1991-12

Numerical investigations of breather solitons in nonlinear vibratory lattices

Walden, Cleon A.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

NUMERICAL INVESTIGATIONS OF BREATHER SOLITONS
IN NONLINEAR VIBRATORY LATTICES

by

Cleon Walden, Jr.

December, 1991

Co-Advisor Bruce Denardo
Co-Advisor Andrés Larraza

Approved for public release; distribution is unlimited.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable) 33	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000
8a NAME OF FUNDING / SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)

NUMERICAL INVESTIGATIONS OF BREATHER SOLITONS IN NONLINEAR VIBRATORY LATTICES

12 PERSONAL AUTHOR(S)
Cleon A. Walden

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	December 1991	95

16 SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defence or the U.S. Government.

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Nonlinear Lattice, Breather Solitons, Numerical Simulation

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Breather solitons in a one-dimensional lattice of coupled nonlinear oscillators are numerically investigated. These are localized nonpropagating steady states that exist at frequencies either below the linear cutoff frequency (corresponding to the extended mode in which all the oscillators are in-phase) or above the upper linear cutoff frequency (corresponding to the extended mode in which each oscillator is 180° out-of-phase with its immediate neighbors). The lattice is damped and parametrically driven. A nonlinear Schrödinger theory, which assumes a modulational amplitude that is weakly nonlinear and slowly varying in space, is compared to the numerical data. The error is roughly 5% at low amplitudes and 20% at high amplitudes. The regions in the drive parameter plane (amplitude vs frequency) where the breathers exist are numerically determined and compared to theory. A substantial discrepancy occurs at lower drive amplitudes where the theory predicts that the lower cutoff breather should exist, but where an instability is observed. Also in contrast to the theory, the region of the upper cutoff breather has relatively large areas in which quasiperiodicity occurs or the motion decays to rest. Quasiperiodicity is also observed in the lower cutoff breather. Finally, instead of a global parametric drive, an end drive is investigated. It is found that, for drive frequencies outside the linear propagation band, there is an amplitude threshold for the periodic ejection or "shedding" of propagating breather solitons. The quasiperiodicity that occurs for a global parametric drive may be a consequence of soliton shedding.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL B. Denardo	22b TELEPHONE (Include Area Code) (408)646-3485 22c OFFICE SYMBOL Ph-de

DD Form 1473, JUN 86 *Previous editions are obsolete*

S/N 0102-LF-014-6603

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

Approved for public release; distribution is unlimited.

Numerical Investigations of Breather Solitons in Nonlinear Vibratory Lattices

by

Cleon A. Walden, Jr.
Lieutenant, United States Navy
B.S., United States Naval Academy, 1986

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING ACOUSTICS

from the

NAVAL POSTGRADUATE SCHOOL
December 1991

ABSTRACT

Breather solitons in a one-dimensional lattice of coupled nonlinear oscillators are numerically investigated. These are localized nonpropagating steady states that exist at frequencies either below the linear cutoff frequency (corresponding to the extended mode in which all the oscillators are in-phase) or above the upper linear cutoff frequency (corresponding to the extended mode in which each oscillator is 180° out-of-phase with its immediate neighbors). The lattice is damped and parametrically driven. A nonlinear Schrödinger theory, which assumes a modulational amplitude that is weakly nonlinear and slowly varying in space, is compared to the numerical data. The error is roughly 5% at low amplitudes and 20% at high amplitudes. The regions in the drive parameter plane (amplitude vs frequency) where the breathers exist are numerically determined and compared to theory. A substantial discrepancy occurs at lower drive amplitudes where the theory predicts that the lower cutoff breather should exist, but where an instability is observed. Also in contrast to the theory, the region of the upper cutoff breather has relatively large areas in which quasiperiodicity occurs or the motion decays to rest. Quasiperiodicity is also observed in the lower cutoff breather. Finally, instead of a global parametric drive, an end drive is investigated. It is found that, for drive frequencies outside the linear propagation band, there is an amplitude threshold for the periodic ejection or "shedding" of propagating breather solitons. The quasiperiodicity that occurs for a global parametric drive may be a consequence of soliton shedding.

C.1

TABLE OF CONTENTS

I. INTRODUCTION

A. BACKGROUND

A soliton is a localized wave whose shape or envelope is constant as a result of nonlinearities and dispersion. Furthermore, it collides elastically with other waves. Solitons were first observed as canal surface waves by John Scott Russell in 1834 (Dodd et al. 1978). Other systems are now known to support solitons of various types. Research in fiber optic solitons is currently very active because of their potential use in communications and optical switching (Mollenauer et al. 1991).

In this thesis, a one-dimensional lattice of coupled nonlinear oscillators is numerically investigated. This system is known to possess kink solitons (Galvin 1990, Denardo et al. 1991). It will be shown that breather solitons can also exist. These are nonpropagating steady states in which most of the oscillators are approximately at rest while those in a localized region have substantial amplitude. Such a state is clearly not a linear mode of the system. Breathers are self-trapped states that occur at frequencies outside the linear propagation band. Two types exist: lower cutoff breathers, in which the oscillators are in-phase, and upper cutoff breathers, in which the oscillators are 180° out-of-phase. Lower and upper cutoff solitons can be considered as finite-amplitude modulations of the uniform lower and upper cutoff modes, respectively.

In the absence of drive and dissipation, it is known that weakly nonlinear breathers described by the ϕ^4 theory are unstable, although the decay rate is extremely slow (Segur and Kruskal 1987). It is also known that instabilities can occur as a result of discreteness in systems that are undriven and undamped (Goedde 1990). In this thesis, dissipation and global drive are employed, which lead to stable breathers.

The simplicity of the model equation suggests that breathers can occur in a variety of systems. Indeed, the first observation of breathers was as cross surface waves on a long channel of liquid (Wu et al. 1984). Lower cutoff breathers have been observed in a pendulum lattice (Denardo 1990). Recently, upper cutoff breathers have been observed in a magnetically coupled pendulum lattice and confirmed numerically with an equation that models this system (Atchley 1991).

It is also shown that, for a local pure-frequency drive confined to one end site of a lattice, it is possible to shed propagating solitons. The shedding occurs at a frequency that is quasiperiodic relative to the drive frequency. This phenomenon has previously been observed in only one system, surface waves just below the second cutoff mode in a large tank (Kit et al. 1987, Shemer 1990). The existence of the soliton shedding in a simple lattice suggests that it is a general phenomenon which can occur in a variety of systems, and that a simple fundamental explanation should exist, although no such explanation is known at present.

B. EQUATION OF MOTION

To motivate the equation of motion that will be investigated, a model system is considered. The system is a uniform lattice of linearly coupled simple pendulums that oscillate transverse to the lattice. The torque on the nth pendulum is:

$$\tau_n = \mu \left(\theta_{n+1} - 2\theta_n + \theta_{n-1} \right) - mgl \sin(\theta_n), \qquad I.B.1$$

where μ is the torsional constant that characterizes the coupling, m is the pendulum mass, and l is the pendulum length. For weakly nonlinear motion, the approximation

$$\sin(\theta) \doteq \theta - \frac{1}{6}\theta^3 \qquad I.B.2$$

$$\sin(\theta) \doteq \theta - \frac{1}{6}\theta^3 \qquad I.B.2$$

can be made. Linear damping is included by adding to I.B.1 a torque proportional to the velocity θ'_n (with a negative coefficient), where the prime denotes differentiation with respect to time. To sustain localized structures in the lattice, a global parametric drive that results from vertically oscillating the lattice, is utilized. By the Equivalence Principle, in the reference frame of the lattice the effective acceleration due to gravity is:

$$g_{eff} = g + a\cos(2\omega t), \qquad I.B.3$$

where a is the acceleration amplitude of the drive and 2ω is the drive frequency.

Combining the above effects, and using Newton's Second Law, the equation of motion of for the lattice can be expressed as:

$$\theta''_n - c^2(\theta_{n+1} - 2\theta_n + \theta_{n-1}) + \beta\theta'_n + [\omega_o^2 + 2\eta\cos(2\omega t)]\theta_n = \alpha\theta_n^3, \quad I.B.4$$

where c^2 is a measure of the coupling between lattice sites and ω_o is the cutoff frequency of a pendulum. For simplicity, the term proportional to the product of the drive and nonlinearity has been dropped. It can be shown that this has no essential effect upon the localized structures. The nonlinear coefficient α equals $\omega_o^2/6$ if (I.B.4) is to approximate a pendulum lattice. The model is generalized by allowing the nonlinear coefficient to be positive or negative. The lattice is driven parametrically with an amplitude of 2η. The fundamental response frequency ω is half the drive frequency. The lower cutoff mode of the lattice is characterized by uniform motion in the lattice (Fig. I.B.1), and has linear frequency ω_o. The upper cutoff mode has a 180° phase difference between adjacent lattice sites (Fig. I.B.2), and has linear frequency $(\omega_o^2 + 4c^2)^{1/2}$. For

a softening ($\alpha > 0$) lattice, the upper cutoff mode is stable. The lower cutoff mode is subject to the Benjamin-Feir instability (Denardo 1990), and the motion evolves into one or more breather solitons (Fig. I.B.3). For a hardening ($\alpha < 0$) lattice, the roles of the cutoff modes are reversed and an upper cutoff breather evolves (Fig. I.B.4).

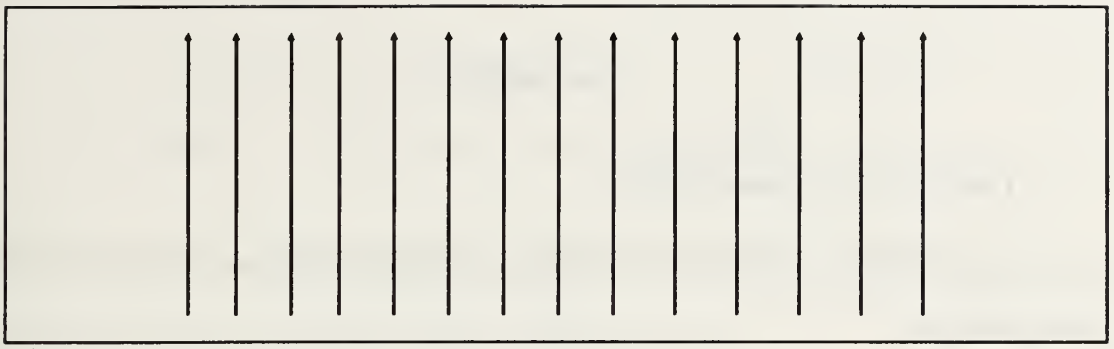

Figure I.B.1 Lower Cutoff Mode

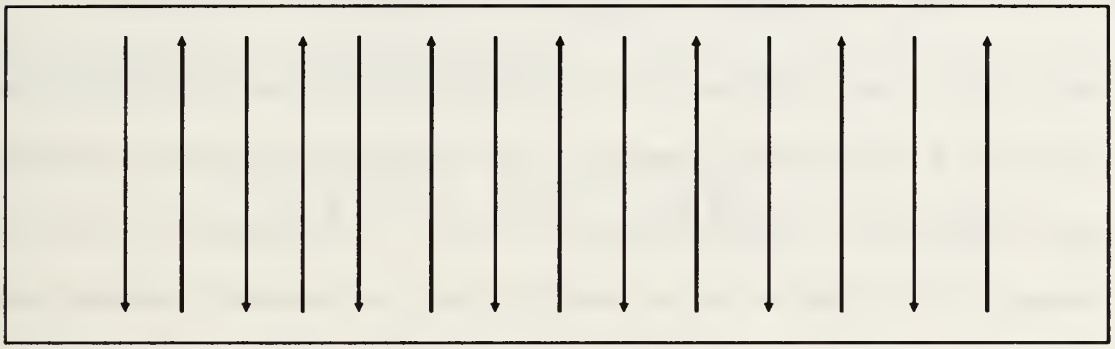

Figure I.B.2 Upper Cutoff Mode

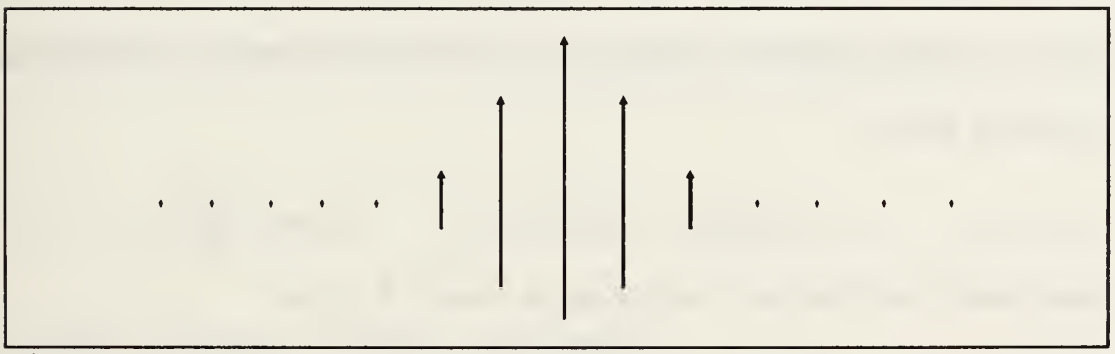

Figure I.B.3 Lower Cutoff Breather

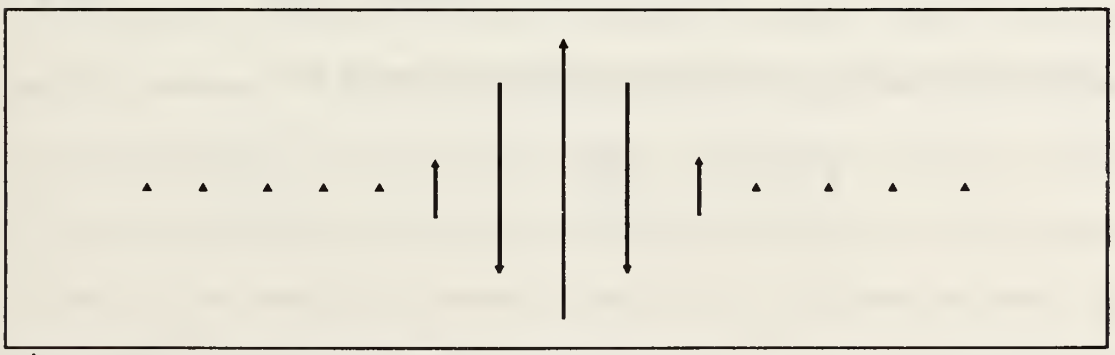

Figure I.B.4 Upper Cutoff Breather

II. THEORY

A. LOWER CUTOFF BREATHER

For modulations of the lower cutoff mode, the displacement (Θ_n) in I.B.4 can be written approximately as:

$$\theta_n = A(n,t)\, e^{i\omega t} + \text{complex conjugate}, \qquad II.A.1$$

where A(x,t) is a complex differentiable function of its arguments. The lattice spacing is assumed to be unity. A displacement (θ_n) containing a third harmonic term produced approximately the same results as II.A.1 and was consequently discarded. Substituting II.A.1 into I.B.4 and requiring that A be slowly varying and weakly nonlinear, one obtains the modulational equation

$$2i\omega A - c^2 A_{xx} + (\omega_o^2 - \omega^2 + i\omega\beta)A + \eta A^* = 3\alpha |A|^2 A, \qquad II.A.2$$

which is a nonlinear Schrödinger equation. If the modulation is assumed to be nonpropagating, a stationary solution

$$A(x,t) = A(x)\, e^{i\delta} \qquad II.A.3$$

exists where A and δ are real. Substituting II.A.3 into II.A.2 gives:

$$-c^2 A'' + (\omega_o^2 - \omega^2)A - 3\alpha A^3 + i\omega\beta A = -\eta A e^{-2i\delta}. \qquad II.A.4$$

Equating the imaginary parts of both sides yields an expression for δ:

$$\sin(2\delta) = \frac{\omega\beta}{\eta}. \qquad II.A.5$$

The real part of II.A.4 yields:

$$c^2 A'' - \mu A + 3 \alpha A^3 = 0, \qquad \text{II.A.6}$$

$$\text{where} \qquad \mu = \omega_o^2 - \omega^2 + \sqrt{\eta^2 - \omega^2\beta^2}\,.$$

There are actually two possibilities because of the radical. However, for a softening lattice the negative branch leads to an unstable solution (Denardo 1990).

Equation II.A.6 can be integrated to yield an elliptic function. A localized solution to II.A.6 is of the form A = a sech(bx), provided

$$a = \sqrt{\frac{2c^2}{3\alpha}}\, b \quad \text{and} \quad b = \sqrt{\frac{\mu}{c^2}}\,.$$

$$\text{Then} \qquad A = \sqrt{\frac{2\mu}{3\alpha}}\ \text{sech}\left[\sqrt{\frac{\mu}{c^2}}\ (x - x_o)\right]. \qquad \text{II.A.7}$$

Substituting the solution for A into II.A.1, the expression for the displacement is:

$$\theta_n = 2\sqrt{\frac{2\mu}{3\alpha}}\ \text{sech}\left[\sqrt{\frac{\mu}{c^2}}\ (x - x_o)\right] \cos(\omega t + \delta). \qquad \text{II.A.8}$$

This solution is valid for $\mu > 0$ and $\alpha > 0$ (softening).

The physical reasoning for the existence of the breather is derived from the curvature of the modulation envelope. For the softening lower cutoff breather, negative curvature in the body of the envelope produces a restoring force (Fig. II.A.1). The frequency in the body is below cutoff due to the nonlinearity which decreases the frequency more than the curvature increases it. In the tail of the envelope the curvature is positive, thus an anti-restoring force. The frequency is below cutoff and the breather is evanescent in the tail. At the inflection points, the

nonlinearity is the sole reason the frequency is below cutoff. The breather is thus a self-trapped state due to the combination of the nonlinearity and curvature of the modulation. As a result, steady state motion is possible. This motion has been observed in the simulated lattice.

Concluding that steady state motion is possible, one should be able to determine the values of η and ω where a stable solution exists. Requiring μ from II.A.6 to be positive,

$$\eta^2 > (\omega_o^2 - \omega^2)^2 + \omega^2 \beta^2. \qquad II.A.9$$

If $\omega \approx \omega_o$, this yields the Matthieu hyperbola. The sech solution (Eqn. II.A.8) is unstable inside the hyperbola because of growth in the wings. There are two conditions for the solution to exist: $\eta > \omega\beta$ and $\mu > 0$. The first condition is $\eta > \omega_o\beta$ if $\omega \approx \omega_o$. The second condition is related to II.A.9, which can be rewritten as

$$\sqrt{\eta^2 - \omega^2 \beta^2} > |\omega_o^2 - \omega^2|. \qquad II.A.10$$

If $\omega > \omega_o$, then $\mu > 0$. However, if $\omega < \omega_o$ the requirement for $\mu > 0$ is $\eta > \omega\beta$. The sech solution decays to rest below the boundary $\eta > \omega\beta$.

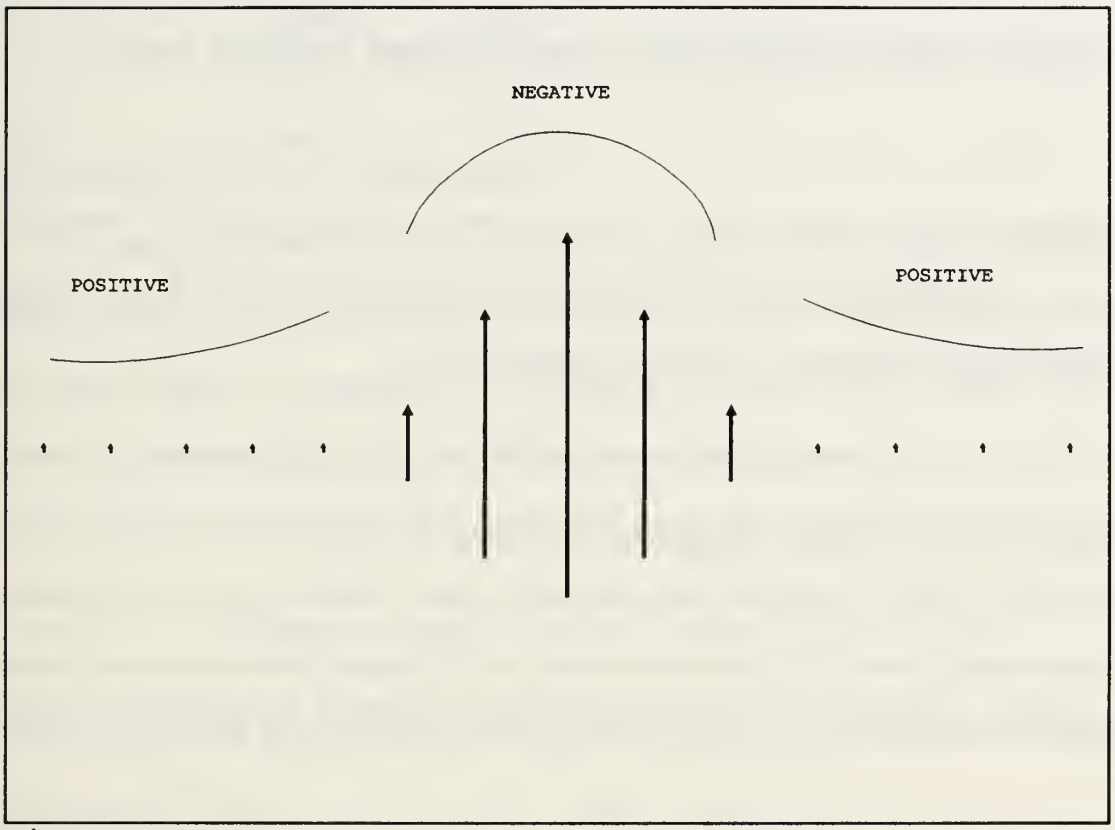

Figure II.A.1 Lower Cutoff Breather Envelope Curvatures

B. UPPER CUTOFF BREATHER

The displacement (Θ_n) for modulations of the upper cutoff mode can be written as

$$\theta_n = (-1)^n A(n,t) e^{i\omega t} + \text{complex conjugate} \qquad II.B.1$$

Where A(n,t) is a complex differentiable function of its arguments. Again, the higher order terms have been neglected, and the lattice spacing is assumed to be unity. The equation that describes the weakly nonlinear complex amplitude is again a nonlinear Schrödinger equation:

$$2i\omega A_t + c^2 A_{xx} + (\omega_1^2 - \omega^2 + i\omega\beta)A + \eta \bar{A} = 3\alpha |A|^2 A. \qquad II.B.2$$

Equation II.B.2 is identical to II.A.2 with the substitutions $\omega_o^2 \rightarrow \omega_1^2$ and $c^2 \rightarrow -c^2$, where $\omega_1^2 = \omega_0^2 + 4c^2$, which is the square of the linear frequency of the upper cutoff mode. Assuming a nonpropagating modulation, a stationary solution for A is:

$$A(x,t) = A(x)\, e^{i\delta}, \qquad II.B.3$$

where A(x) and δ are real. Substituting II.B.3 into II.B.2 gives:

$$c^2 A'' + (\omega_1^2 - \omega^2)A - 3\alpha A^3 + i\omega\beta A = -\eta A e^{-2i\delta}. \qquad II.B.4$$

Setting the imaginary parts of both sides equal and solving for δ, the phase relation is again:

$$\sin(2\delta) = \frac{\omega\beta}{\eta}. \qquad II.B.5$$

The real part of II.B.4 yields:

$$-c^2 A'' + \nu A + 3\alpha A^3 = 0 \qquad II.B.6$$

$$\text{Where} \qquad \nu = \omega^2 - \omega_1^2 + \sqrt{\eta^2 - \omega^2\beta^2}$$

which is similar to II.A.6. Again, the selected sign of the radical leads to a stable solution. A trial solution of the form A = a sech(bx) used in II.B.4 yields:

$$A = \sqrt{\frac{2\nu}{-3\alpha}}\ sech\left[\sqrt{\frac{\nu}{c^2}}\ (x - x_o)\right]. \qquad II.B.7$$

Substituting the solution for A into II.B.1, the displacement (Θ_n) for the upper cutoff mode is:

$$\theta_n = (-1)^n\, 2\sqrt{\frac{2\nu}{-3\alpha}}\ sech\left[\sqrt{\frac{\nu}{c^2}}\ (x - x_o)\right]\cos(\omega t + \delta), \quad II.B.8$$

which is valid for $\nu > 0$ and $\alpha < 0$ (hardening).

The physical mechanism for the existence of the upper cutoff breather is essentially identical to that of the lower cutoff breather. The negative curvature in the body of the hardening upper cutoff breather now produces an anti-restoring force (Fig. II.B.1) (Denardo 1990), however, the nonlinearity is stronger than the curvature and the frequency in the body is above the linear upper cutoff frequency. The positive curvature in the tails of the modulation creates a restoring force and the frequency is again above the cutoff frequency. Similar to before, the frequency is above the cutoff frequency at the inflection points as a sole result of the nonlinearity. Thus, the upper cutoff breather is also a self-trapped state due to the combination of curvature and nonlinearity.

As for the lower cutoff case, a region of stable, steady state motion should also exist for the upper cutoff breather. Constraining ν from II.B.6 to be positive,

$$\eta^2 > (\omega_1^2 - \omega^2)^2 + \omega^2 \beta^2. \qquad II.B.9$$

Again, if $\omega \approx \omega_o$ this yields the Matthieu hyperbola. The wings of the sech solution are unstable inside the hyperbola. There are two conditions for the solution exist: $\eta > \omega\beta$ and $\nu > 0$. The first condition is $\eta > \omega_o\beta$ if $\omega \approx \omega_o$. The second condition is related to II.A.20 which can be rewritten as

$$\sqrt{\eta^2 - \omega^2 \beta^2} > |\omega_1^2 - \omega^2|. \qquad II.B.10$$

If $\omega > \omega_1$, then $\mu > 0$. However, if $\omega < \omega_1$ the requirement for $\mu > 0$ is $\eta > \omega\beta$. The sech solution decays to rest below the boundary $\eta > \omega\beta$.

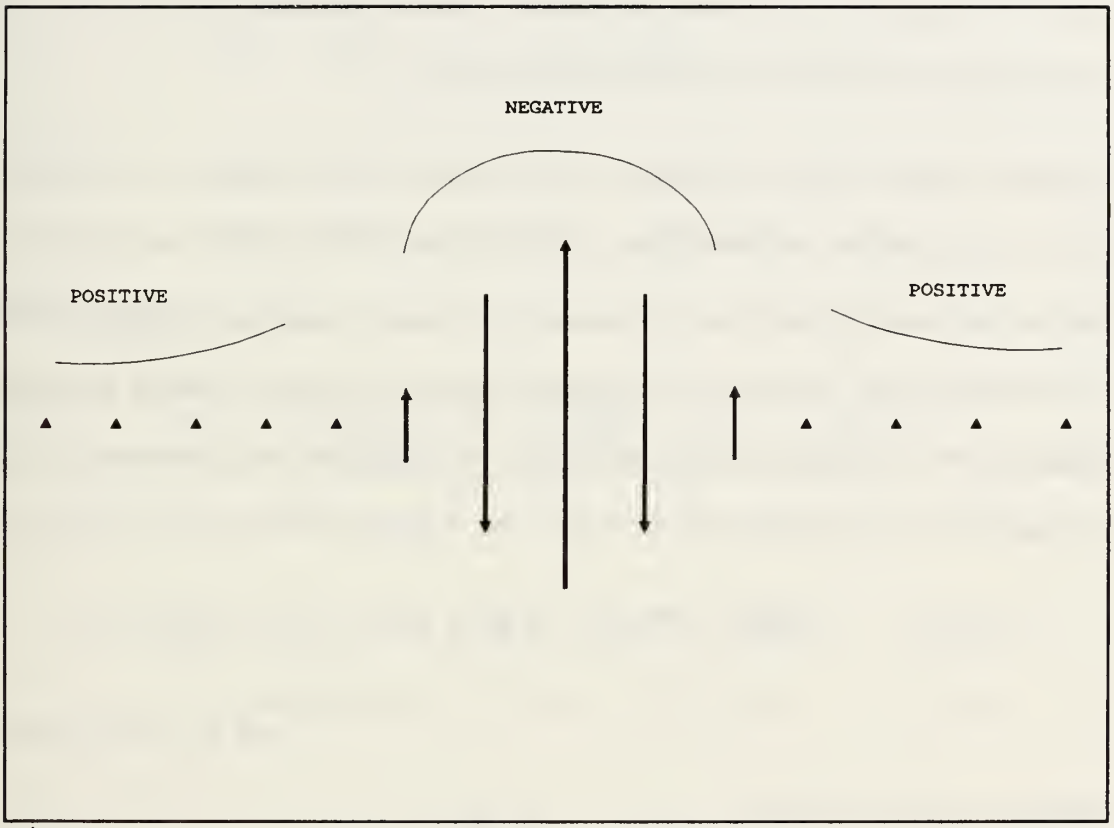

Figure II.B.1 Upper Cutoff Breather Envelope Curvatures

C. END DRIVEN LATTICES

Nonlinear lattices driven at one end and at a frequency outside the linear propagation band have been observed to periodically eject solitons (Sect.III.D). This is in strong contrast to the linear theory which predicts only a steady state evanescent wave, as will now be shown. The equation of motion for a lattice of linearly coupled nonlinear oscillators (I.B.4) can be modified to replace the parametric drive with a direct drive. Further restricting the drive to the end site alone and removing the periodic boundary condition yields:

$$n=0: \quad \theta_0'' - c^2(\theta_1 - \theta_0) + \beta\theta_0' + \omega_o^2\,\theta_0 + \eta\cos(\omega t) - \alpha\theta_0^3 \qquad II.C.2$$

$$n>0: \quad \theta_n'' - c^2(\theta_{n+1} - \theta_n) + \beta\theta_n' + \omega_o^2\,\theta_n - \alpha\theta_n^3 .$$

This produces an end driven lattice. The lattice is still linearly coupled and nonlinear, but is now more suitable for the observation of propagating solitons which have a spatially varying phase (Denardo 1990). Focusing on the linear motion in the continuum limit, equations II.C.2 have the form:

$$x > 0: \quad y_{tt} - c^2 y_{xx} + \beta y_t + \omega_o^2 y = 0 \qquad II.C.3$$

$$x = 0: \quad y_{tt} - \frac{c^2}{a} y_x + \omega_o^2 y + \eta\cos(\omega t) .$$

Consider a solution of the form:

$$y = A\, e^{ikx - i\omega t - \alpha x} + \text{complex conjugate}, \qquad II.C.4$$

with a complex wavenumber $\kappa = k + i\alpha$. Substituting II.C.4 into II.C.3 yields:

$$-\omega^2 - c^2(ik - \alpha)^2 - i\omega\beta + \omega_o^2 = 0. \qquad II.C.5$$

Separating the real and imaginary parts of II.C.5 gives:

$$Re: \qquad \alpha^2 - k^2 = \frac{\omega_o^2 - \omega^2}{c^2}$$

$$Im: \qquad \alpha\, k = \frac{\omega\beta}{2c^2}.$$

The case of interest is $\omega < \omega_0$. The solution for $\beta = 0$ is:

$$k = 0 \quad and \quad \alpha = \sqrt{\frac{\omega_o^2 - \omega^2}{c^2}}. \qquad II.C.6$$

Physically, the wave is purely exponential in space; the sign of the displacement is the same for every lattice site. If $\beta \neq 0$ then k can be eliminated using the real and imaginary parts of II.C.5 to obtain:

$$\alpha^4 - 2\,\frac{\omega_o^2 - \omega^2}{2c^2}\,\alpha^2 - \left(\frac{\omega\beta}{2c^2}\right)^2 = 0. \qquad II.C.7$$

Solving for α^2 and choosing the positive root , since α was assumed to be real, gives:

$$\alpha^2 = \frac{1}{2c^2}\left[\omega_o^2 - \omega^2 + \sqrt{(\omega_o^2 - \omega^2)^2 + (\omega\beta)^2}\right]. \qquad II.C.8$$

The solution for k is then:

$$k = \frac{\omega\beta}{2c^2}\,\frac{1}{\alpha}. \qquad II.C.9$$

If the solutions for ω^2 and k are specialized to weak damping, $\omega\beta \ll \omega_o^2 - \omega^2$ then:

$$\alpha = \sqrt{\frac{\omega_o^2 - \omega^2}{c^2}} \quad and \quad k = \frac{\omega\beta}{2c}\,\frac{1}{\sqrt{\omega_o^2 - \omega^2}}. \qquad II.C.10$$

The damping gives rise to a spatial phase ($k \neq 0$). Specializing to strong damping, $\omega\beta \gg \omega_o^2-\omega^2$ or near-propagation, yields:

$$\alpha = k = \sqrt{\frac{\omega\beta}{2c^2}} . \qquad II.C.11$$

This is the largest that k/α can be (unity). In general, $0 \leq k/\alpha \leq 1$ (Fig. II.C.1).

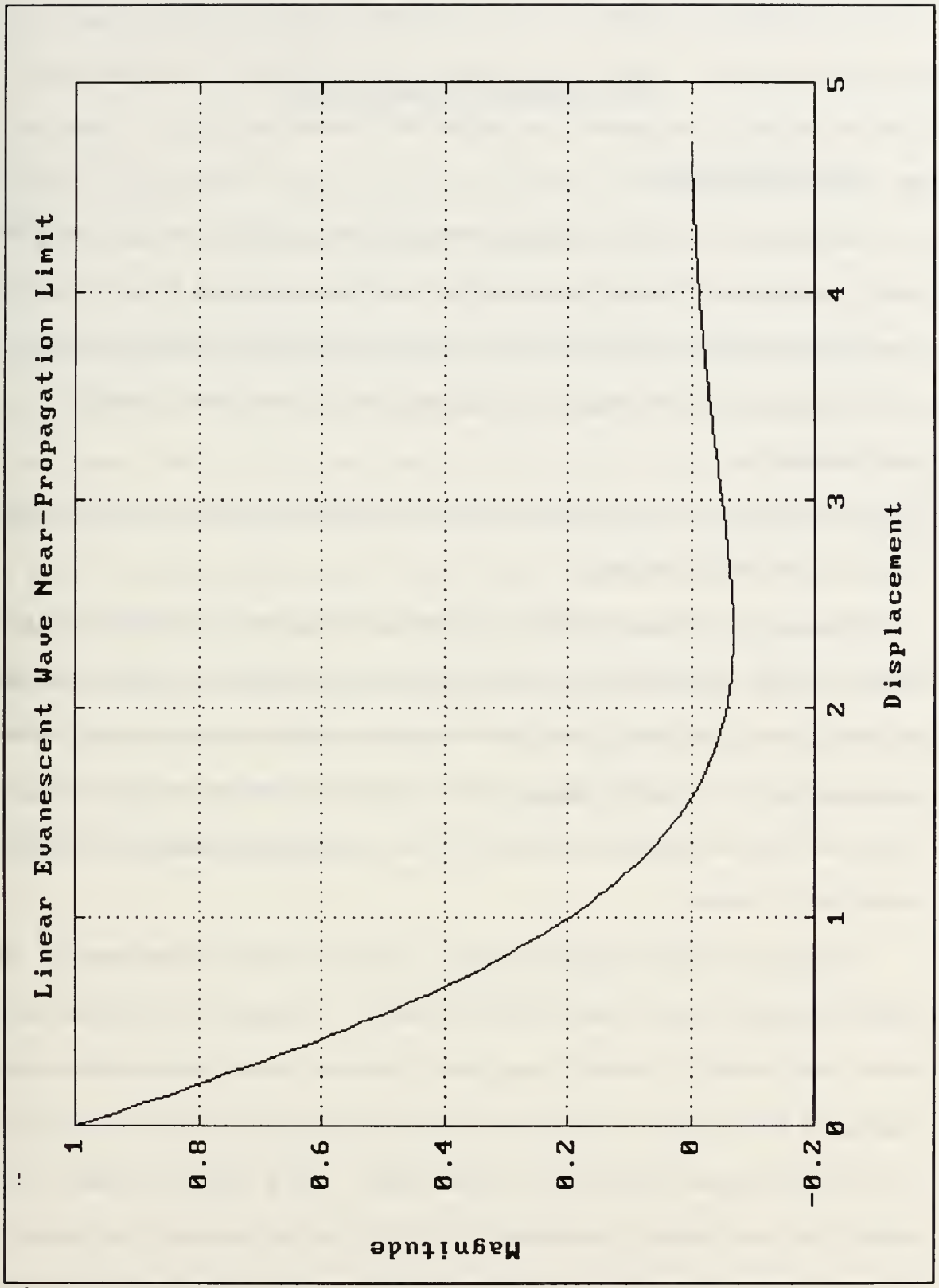

Figure II.C.1 Evanescent Wave For $k/\alpha=1$

III. NUMERICAL SIMULATION

A. IMPLEMENTATION

The motion of the lattice of linearly coupled nonlinear oscillators was solved with the Euler-Cromer method. This implementation utilized acceleration as the basic numerical iteration. The position calculation of each lattice site after a given time step was divided into three sections:

1. The calculation of acceleration for each lattice site due to the position of it and the two adjacent sites.
2. The calculation of the new velocity for each site by multiplying the acceleration by the time step, and adding the old velocity.
3. The calculation of the new position for each lattice site using the new velocity for each site.

Periodic boundary conditions were imposed, thus making the lattice a ring lattice with any curvature ignored. The Euler-Cromer method converged, with an increasing number of time steps, to a "true" lattice amplitude (Fig. III.A.1). A time step that gave an amplitude within .1% of the "true" response amplitude allowed the program to run at a fairly high rate of speed with minimal loss of accuracy.

The program is highly interactive and gives a "real time" picture of the lattice motion. The numerical simulation imposes some interactive restrictions. The elapsed time in the model frame must be reset periodically (preferably every period of the drive) to avoid strong transients when changing the drive frequency. The problem stems from the parametric drive term ($\cos(2\omega t)$) since the time increment is based on the response period. Thus a change in frequency in the middle of the cycle produces a discontinuity in the size of the time increment. The solution is to employ an integer counter to reset the elapsed time after one period of the drive and to require

any frequency changes to occur when the time is reset. The startup of a modulation envelope or "profile" can be accomplished in two ways: restarting a saved profile, or starting a profile created from theory. A profile can be saved while the program is running, thus eliminating the need to duplicate steps required to reach a particular set of parameters. All necessary parameters such as drive amplitude and frequency in addition to the position and velocity for each lattice site are written to a file of the researcher's choice. The modulation must meet certain stability criterion to be recorded and is "captured" at the upper turning point (response) of a selected site. A previously saved profile can be recalled using only the filename. Since the response and drive have a phase difference, the program restarts the profile with a time phase correction to minimize startup transients. A startup file could also be generated using one of the programs that calculate a profile from theory using keyboard input of drive, coupling and other parameters. The equations of motion are in a subroutine, allowing them to be altered or replaced quickly. A different method of calculating the lattice site positions would probably not cause any difficulty as long as the new coordinates were available at the same place in the program as before. A complete list and more detailed description of the program options as well as a program code listing are included in the Appendix.

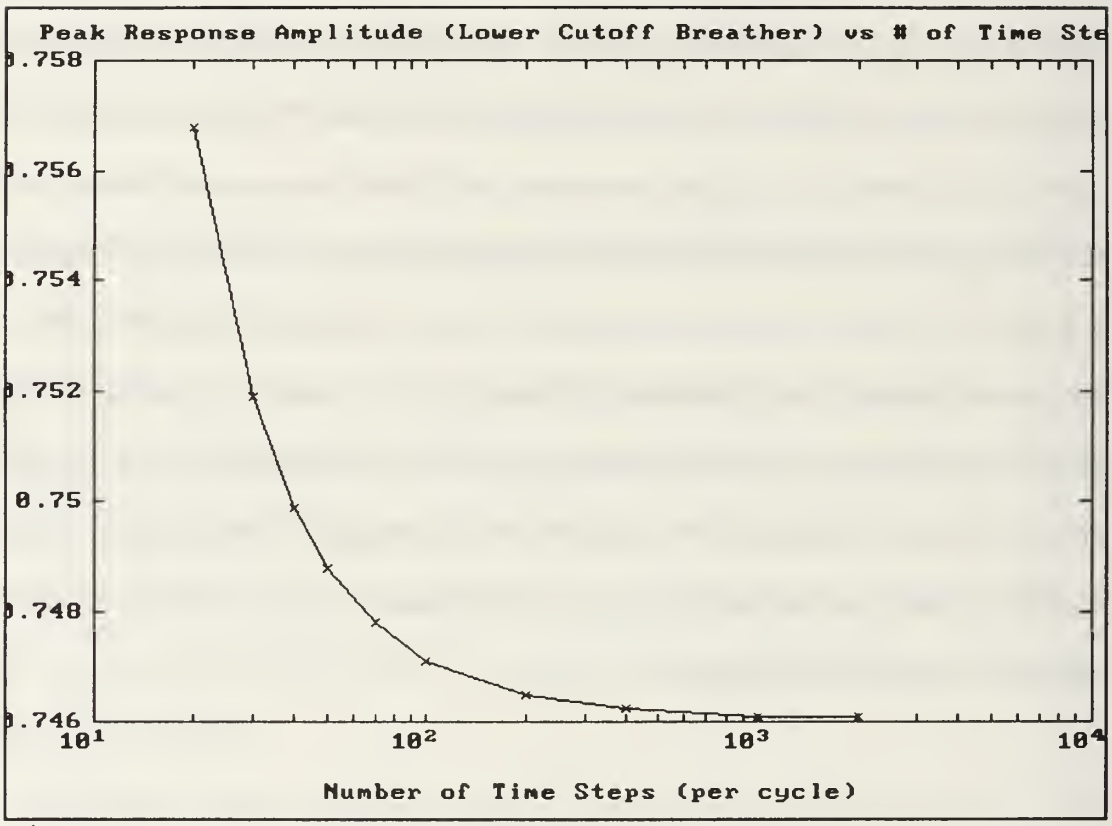

Figure III.A.1 Response Accuracy Plot

B. LOWER CUTOFF BREATHERS

In order to investigate the stability of lower cutoff breathers, a drive plane, consisting of drive amplitudes and frequencies where the particular structure exists, was obtained by numerical solution of the equation of motion. The structure chosen for the lower cutoff drive plane was a symmetric breather (Fig. III.B.1) in which the maximum amplitude is centered on a lattice site (on-site). The nonlinear parameter (α) was 1 (lower cutoff), the coupling (c^2 or γ) was .1 (weak), dissipation (β) was .03 (small) and the lattice consisted of 50 sites long. The points in the drive plane (Fig. III.B.2) were almost exclusively obtained by incrementing the drive amplitude by .1% (or less) and recording the amplitude at which transitions occurred. Additionally, to probe stability, a $\leq \pm 3\%$ (of the maximum response amplitude in the lattice) random kick was applied to the lattice after each increment.

The upper boundary from $\omega = .76$ to .89 was characterized by the lattice "going over the top" after perturbation (random kick). Because, the potential well for the lower cutoff mode (Fig. III.B.3) is not infinitely high, the lattice is able to escape the well (go over the top) and become unstable. There were no visual indications in the lattice of an impending instability before the boundary was reached. The right upper boundary, from $\omega = .89$ to .94, was marked by the generation of another symmetric breather (out of phase with the first) on the opposite side of the lattice ring from the initial breather. The lattice then went over the top from a random site fairly quickly after the generated breather reached an amplitude comparable to the original breather. Further tests with a lattice of 100 sites indicated that the generated breather is approximately 27 sites away from the center of the original breather and will arise on both sides of the original if the sites are available. The breather structure in the extreme lower right corner is quasiperiodic until the boundaries where it goes over the top without the generation of another

breather. The lower boundary from $\omega=.85$ to .947 is preceded by a quasiperiodic region. Ultimately the lattice either goes over the top or decays to rest after very strong transients. Careful observations are suggestive of quasiperiodicity resulting from the energy loss due to soliton shedding. The left lower boundary from $\omega=.76$ to .85 was only preceded by the quasiperiodic region at the points indicated but the strong transients were consistently observed before the lattice went over the top. A comparison of the stable region predicted by theory for the lower cutoff symmetric breather shows good agreement for the upper right boundary but a discrepancy on the lower boundary (Fig. III.B.4). Some destabilizing factor has prevented the lower boundary from reaching the theoretical limit. Experimentation with a half-site centered symmetric breather (Fig. III.B.5) revealed a link between the strong transients observed in the on-site breather drive plane but did not extend the lower boundary to the theoretical limit. The strong transients occur at the same drive amplitudes that the half-site breather shows quasiperiodic behavior. Increasing the drive amplitude results in a transition from half-site to on-site at approximately the start of the quasiperiodic line in the on-site drive plane. The half-site breather is stable as drive amplitude is decreased, but it decays to rest at approximately $\eta=0.9$ for $\omega=0.9$. The transition from steady state motion to rest is interesting in that it shifts back to an on-site breather before it decays to rest.

Comparison of the theoretical profile for the parameters used in the drive plane was done for the lowest and highest amplitude saved profiles as well as a half-site profile. The lowest amplitude comparison was sharper and lower in amplitude than the predicted profile (Fig. III.B.6). The highest amplitude comparison was also sharper and even lower in comparable amplitude (Fig. III.B.7). A comparison of the half-site profile shows much closer agreement with theory although the data still exhibits a greater amplitude (Fig. III.B.8). The amplitude of the saved profile from the program was not exactly the maximum amplitude of that particular state

did not precisely coincide with the peak response amplitude. This was corrected by using the maximum coordinate plus the coordinates for the preceding and following time steps to find the maximum amplitude by parabolic curve fitting. By recording the time step, the data could be scaled according to the time difference between the turning point and actual capture time.

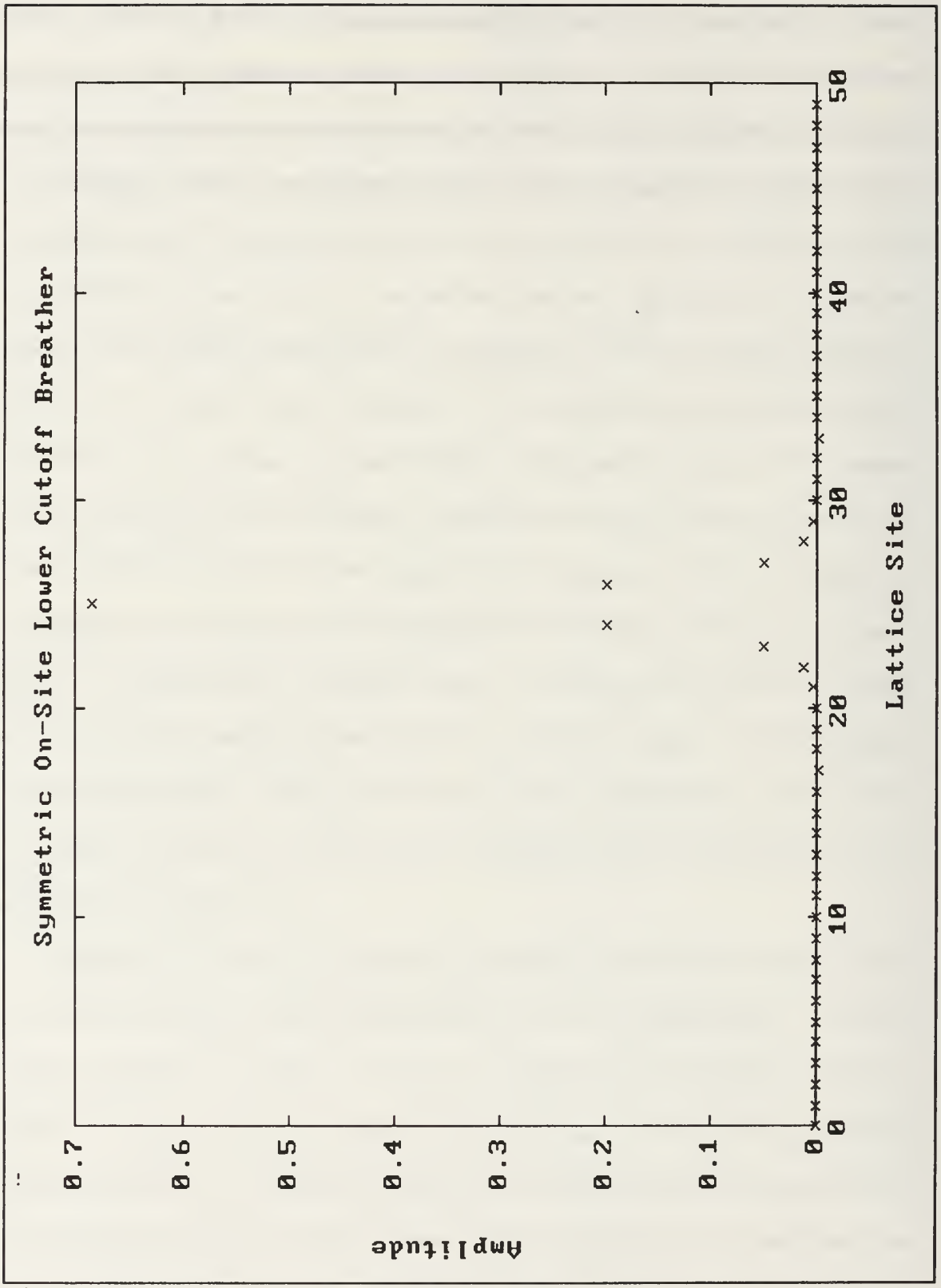

Figure III.B.1 Lower Cutoff Breather

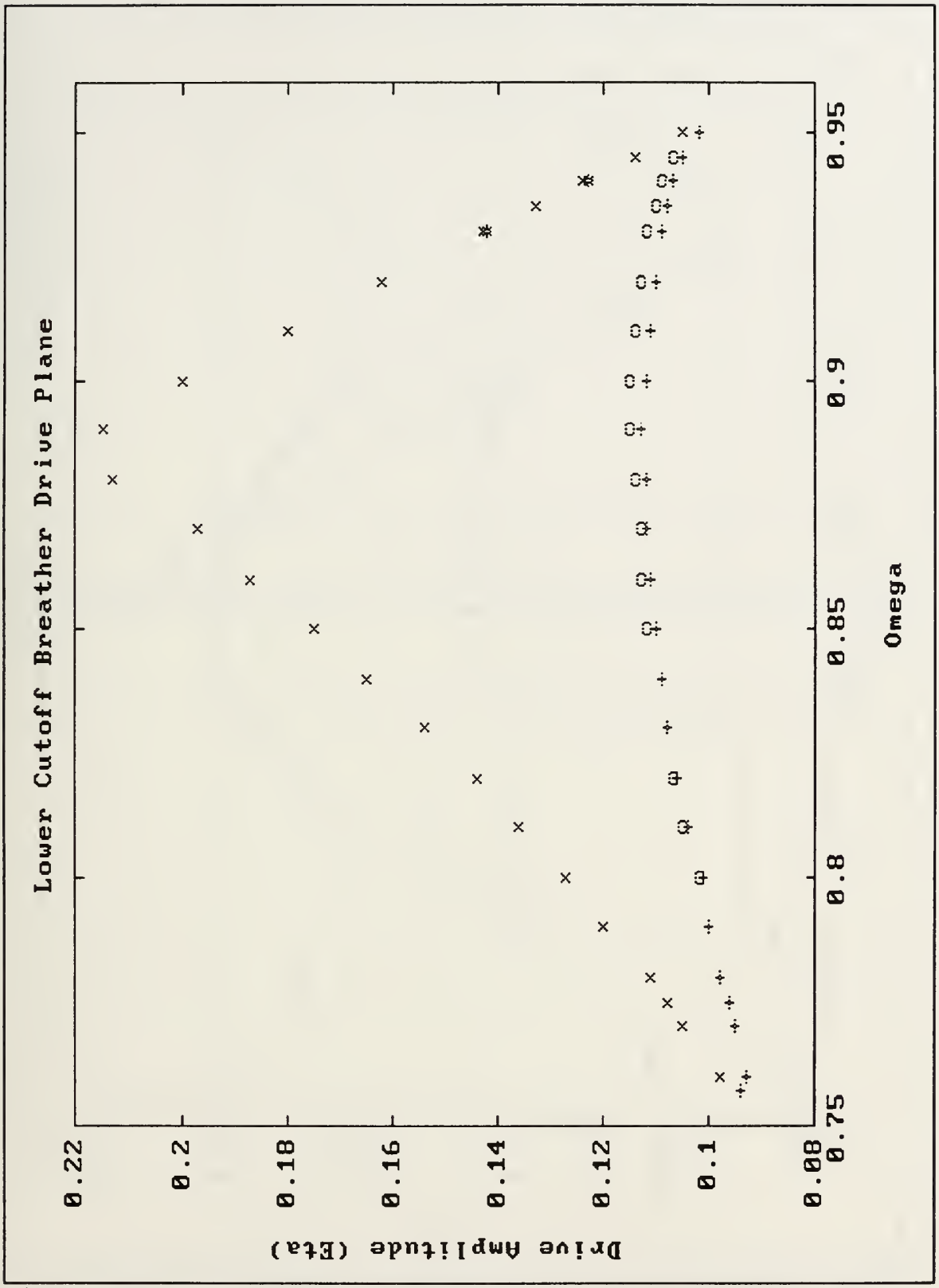

Figure III.B.2 Lower Cutoff Drive Plane

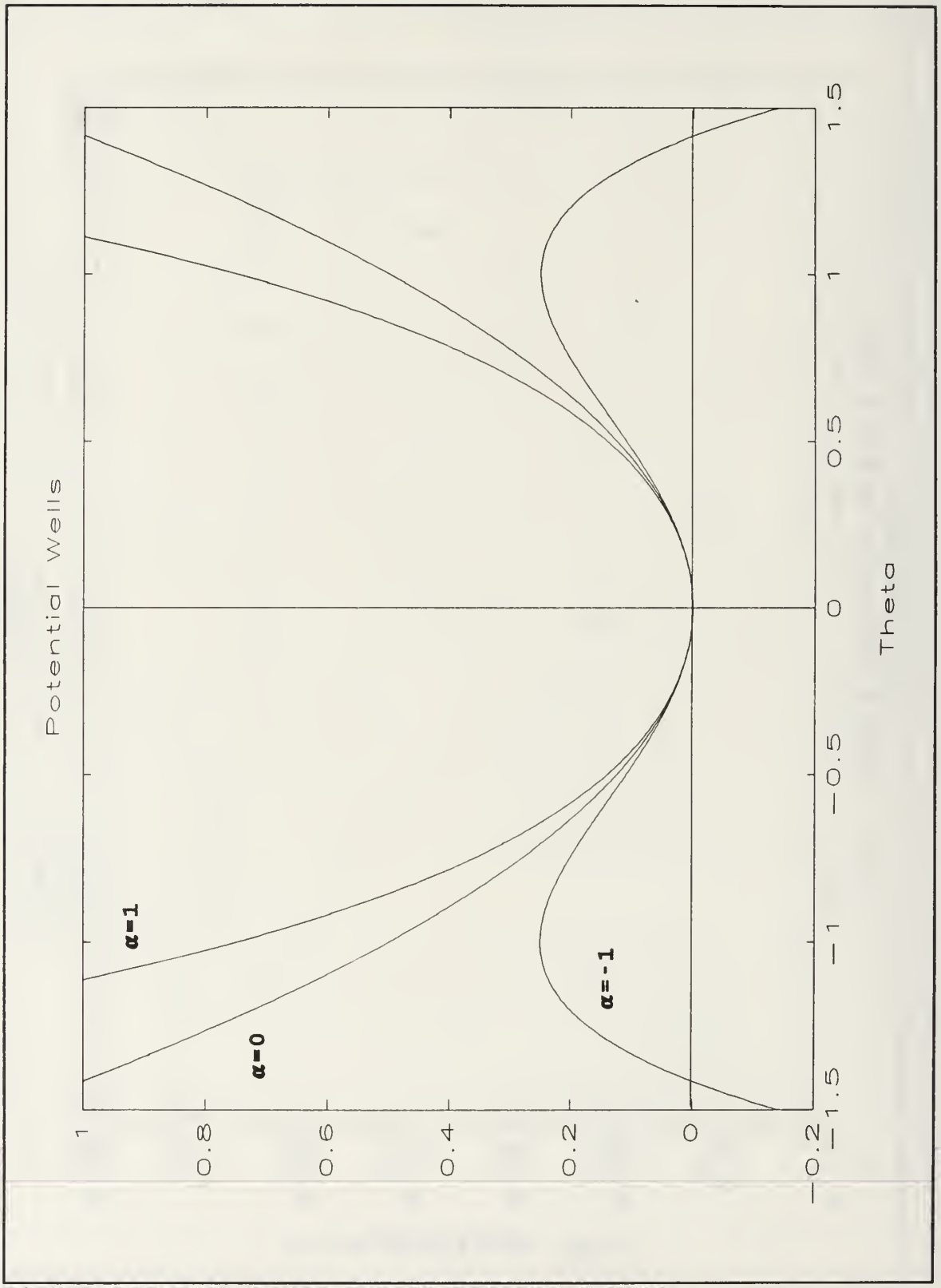

Figure III.B.3 Lower Cutoff Potential Well

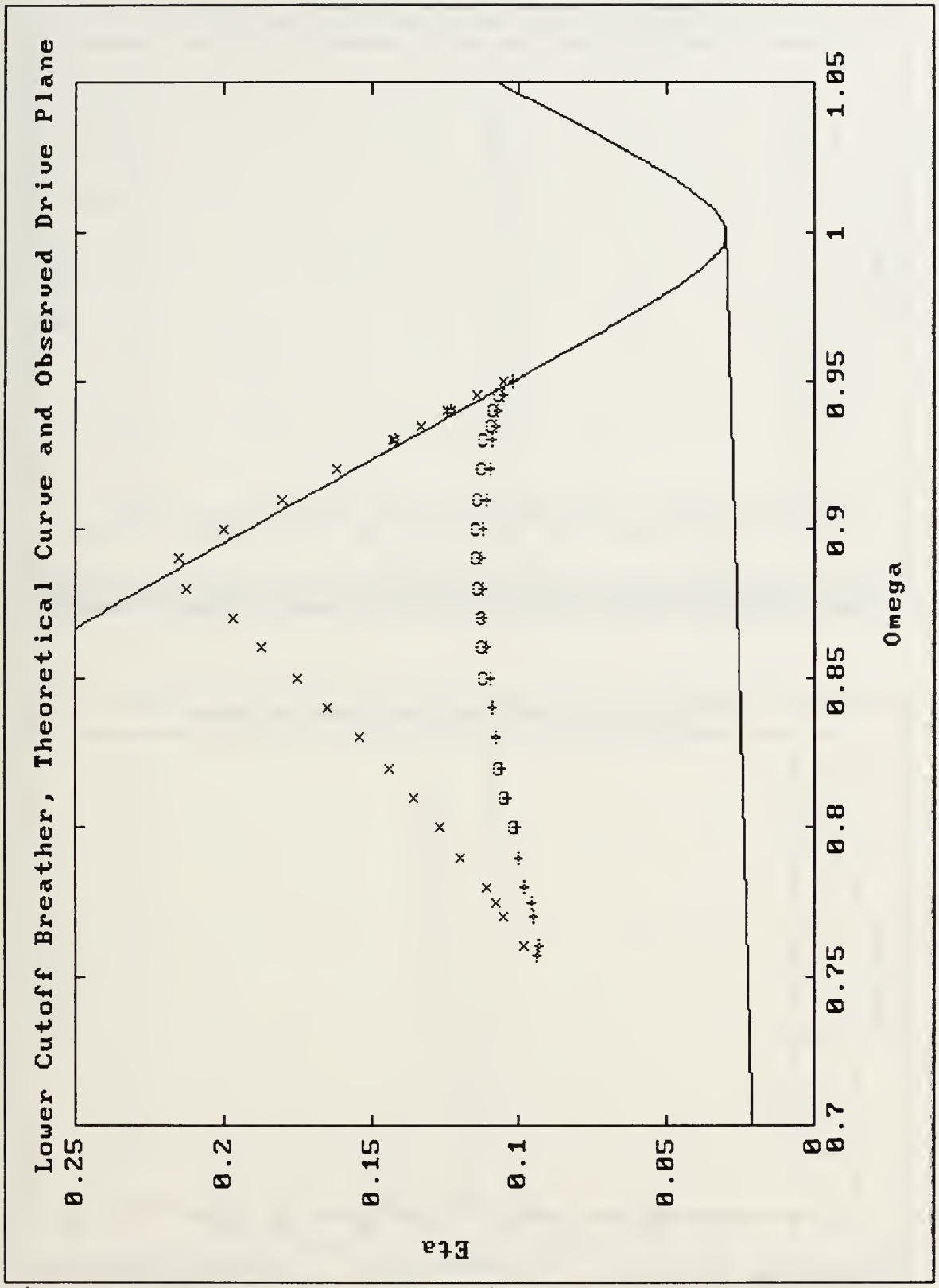

Figure III.B.4 Lower Cutoff Drive Plane and Theory

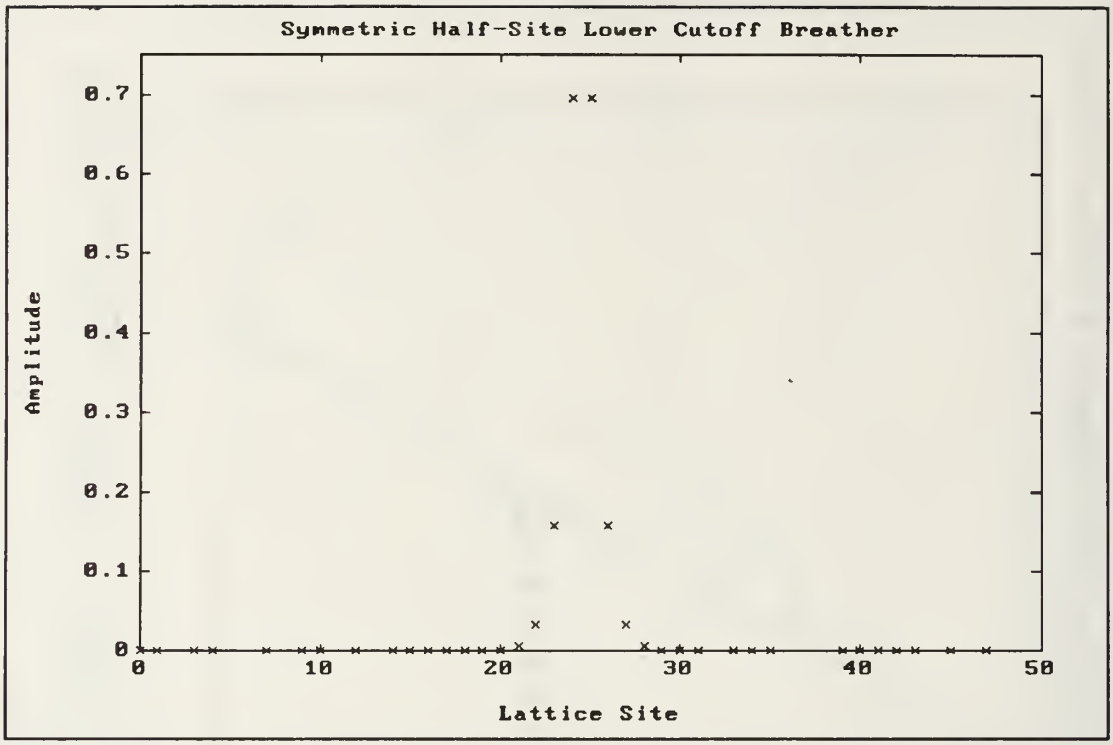

Figure III.B.5 Symmetric Half-Site Lower Cutoff Breather

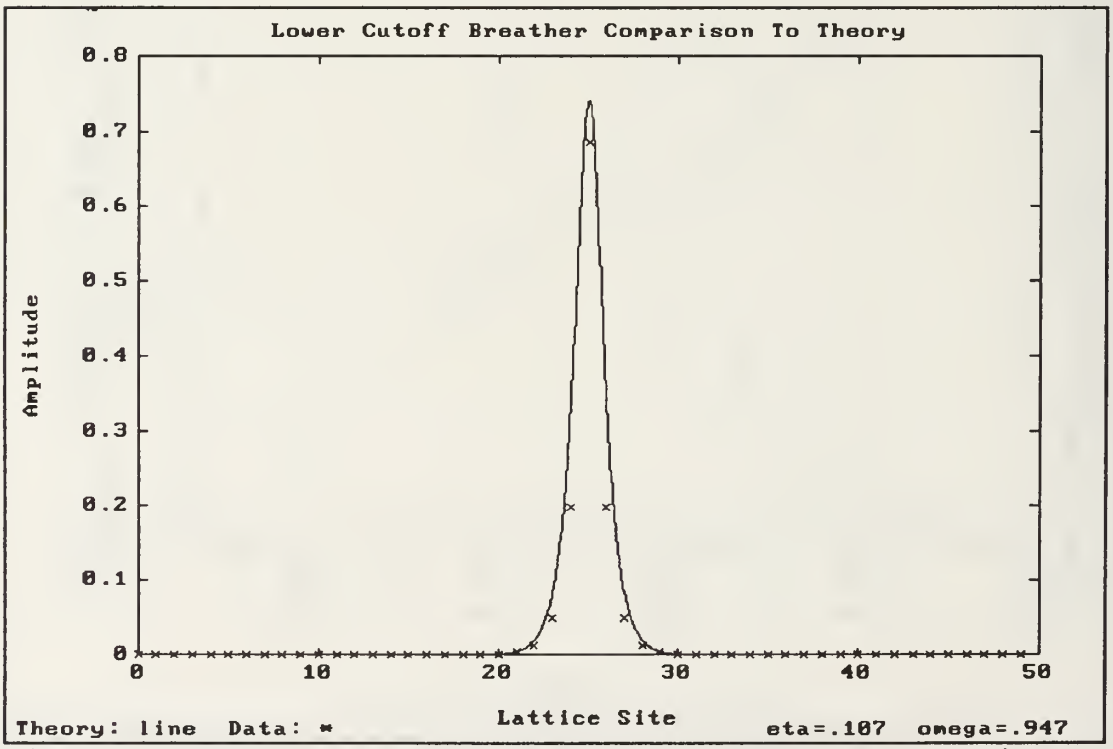

Figure III.B.6 Lower Cutoff Breather Theory Comparison

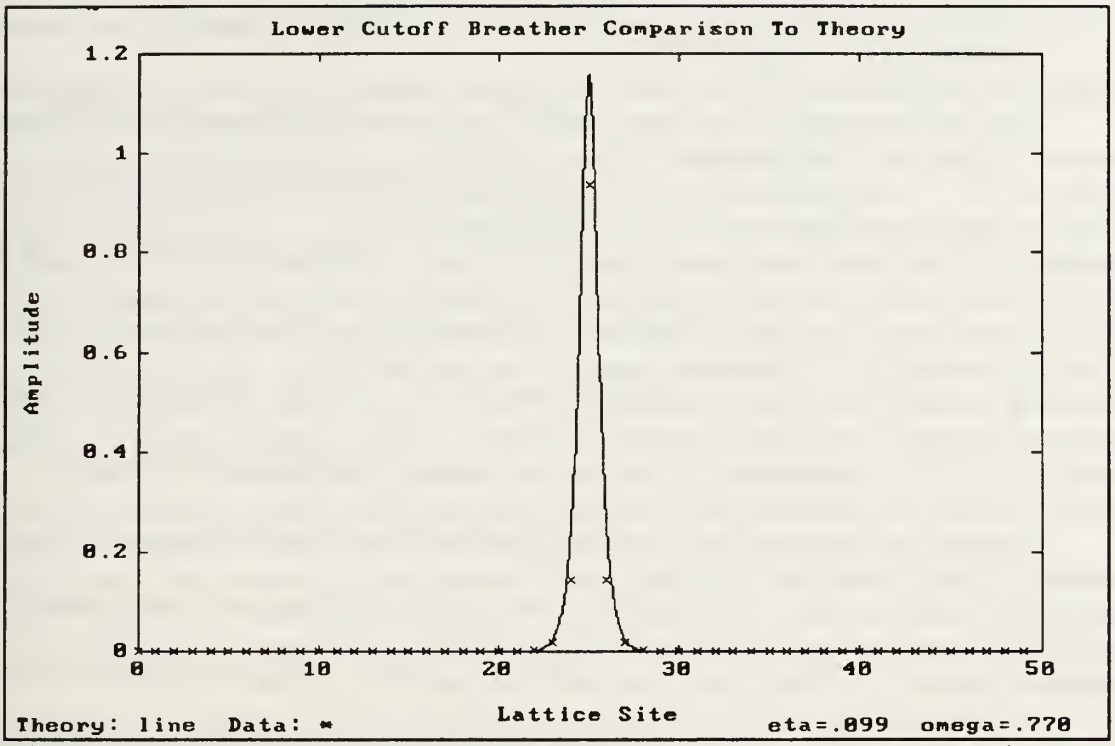

Figure III.B.7 Lower Cutoff Breather Theory Comparison

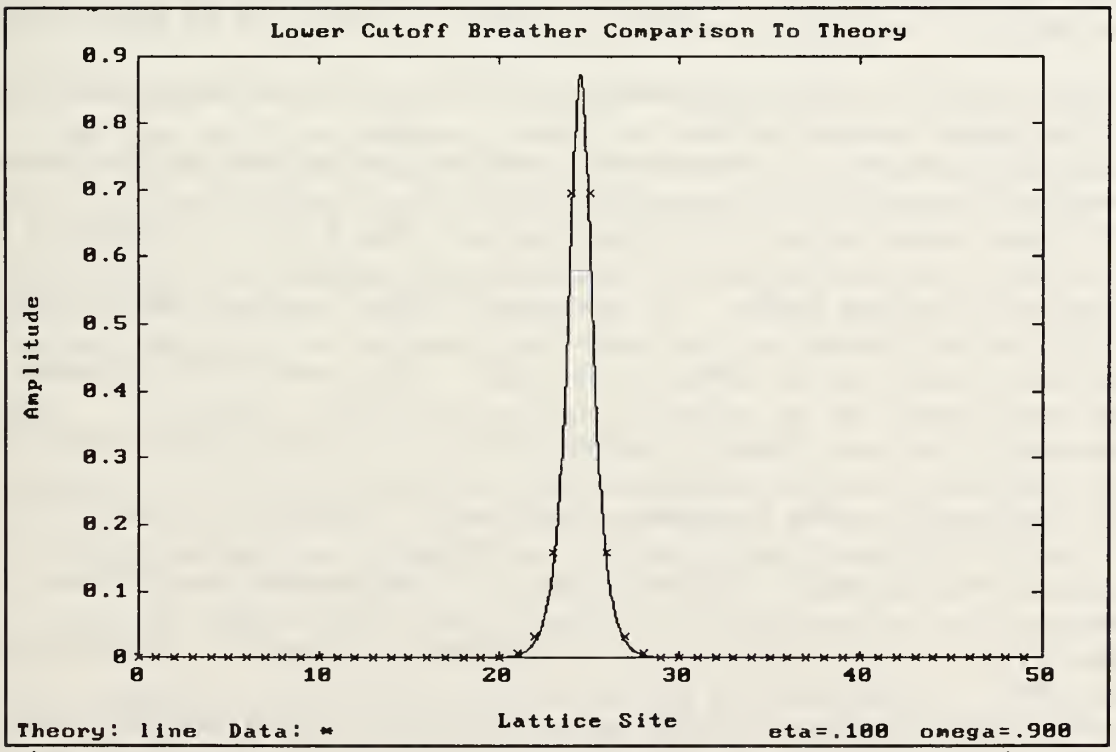

Figure III.B.8 Lower Cutoff Breather Theory Comparison

C. UPPER CUTOFF BREATHERS

The structure chosen for the upper cutoff drive plane was the symmetric on-site breather (Fig. III.C.1). The parameters for the drive plane were α=-1 (hardening), γ=.1 (weak coupling), β=.03 (small) and a lattice length of 50 sites. The points in the drive plane (Fig. III.C.2) were almost exclusively obtained by incrementing the drive amplitude by .001 and recording the amplitude at which instabilities were observed. After each increment, a random kick of $\leq \pm 3\%$ (of the maximum lattice response amplitude) was applied to the lattice.

The upper left boundary marks the sharp transition from a breather structure to a complex upper cutoff mode structure. The transition time was fairly short and was characterized by the growth of one section of the lattice to an amplitude comparable to that of the original breather. The growing section was in a pure upper cutoff mode and spread to include the rest of the lattice after reaching full amplitude. The area inside the circles (Q) was where the lattice motion was quasiperiodic. The boundary was difficult to ascertain for omega 1.25 and higher since local regions appeared to exhibit quasiperiodicity but in fact merely had long settling times. Regions marked H denote areas of no lattice motion. Inside the region and below the lower boundary, the lattice decayed to rest with minimal transients. Region QS was characterized by a quasiperiodic shedding breather. The quasiperiodic breather was usually observed near the boundaries of QC and was similar to Fig. III.C.1. The center site of the breather was quasiperiodic and small solitons were ejected from the center in both directions simultaneously. The quasiperiodic breather spontaneously transitioned to a complex cyclic state at the lower boundary of QC (asterisks). The lattice motion in the small asterisked area (QC) was of two general types, a complex cyclic state or an anti-symmetric half-site breather. The cyclic state that evolved from the quasiperiodic breather appeared to be chaotic and then settled into multiple on-

site breathers. The motion shifted back and forth between the two states with a fairly long cycle. If the lattice was in this particular mode when the boundary to H (right side of QC) was reached, the cycle was broken and the lattice settled into a multiple breather state. The multiple breathers merged two by two until a single on-site breather was left. The single breather then decayed to rest. The lattice also spontaneously shifted from the complex cycle to a half-site antisymmetric breather (Fig. III.C.3). A half-site breather consistently evolved from an quasiperiodic on-site breather at the upper boundary of QC. The half-site breather exhibited a degeneration similar to that of the complex cyclic at the boundary to region H. The breather spontaneously changed back into an on-site breather and then decayed to rest. Multiple antisymmetric half-site breathers were also observed. A state with two half-sites 10-15 sites apart (center-to-center) were believed to be transferring energy by soliton shedding. A growth in amplitude in the center sites of one breather corresponded to the arrival of a soliton from the other breather and the expected decrease in the center-site amplitude of the second breather due to soliton ejection was observed.

The upper and lower boundaries compare well with the theoretical boundaries (Fig. III.C.4). The upper boundary does not quite follow the hyperbola at the higher drive amplitudes, however the theory is approximate at these amplitudes. The lower boundary, which corresponds to the lattice decaying to rest, coincides almost exactly with the predicted threshold of lattice motion. Comparison of theoretical modulation envelopes and the numerical data were completed for extremely high, high and low response amplitude cases as well as a half-site case. The observed amplitudes were adjusted to that of the turning point as described in Section II.B. The normal 180° phase difference between lattice sites was eliminated for easier modulation envelope comparison. The extreme case was not as sharp as predicted and was $\approx 70\%$ of the theoretical amplitude (Fig. III.C.5). Similarly, the amplitude of the high case was $\approx 90\%$ of

theoretical but the envelope was sharper than expected (Fig. III.C.6). The low amplitude case exhibited the inverse characteristics of the extreme case (Fig. III.C.7). The observed envelope was sharper than the theory and $\approx 5\%$ higher in amplitude. A comparison of the half-site case showed very good agreement with theory (Fig. III.C.8).

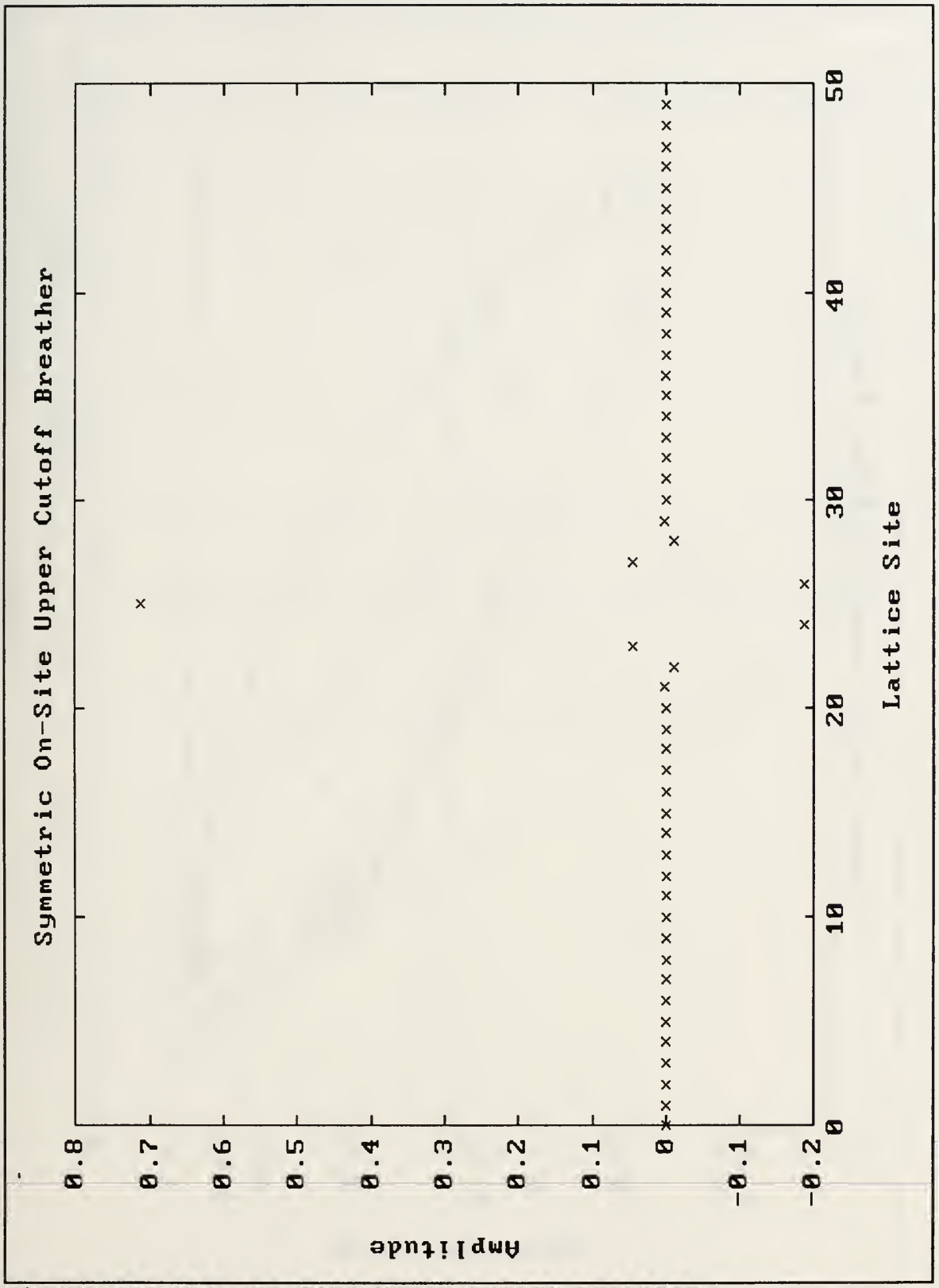

Figure III.C.1 On-Site Upper Cutoff Breather

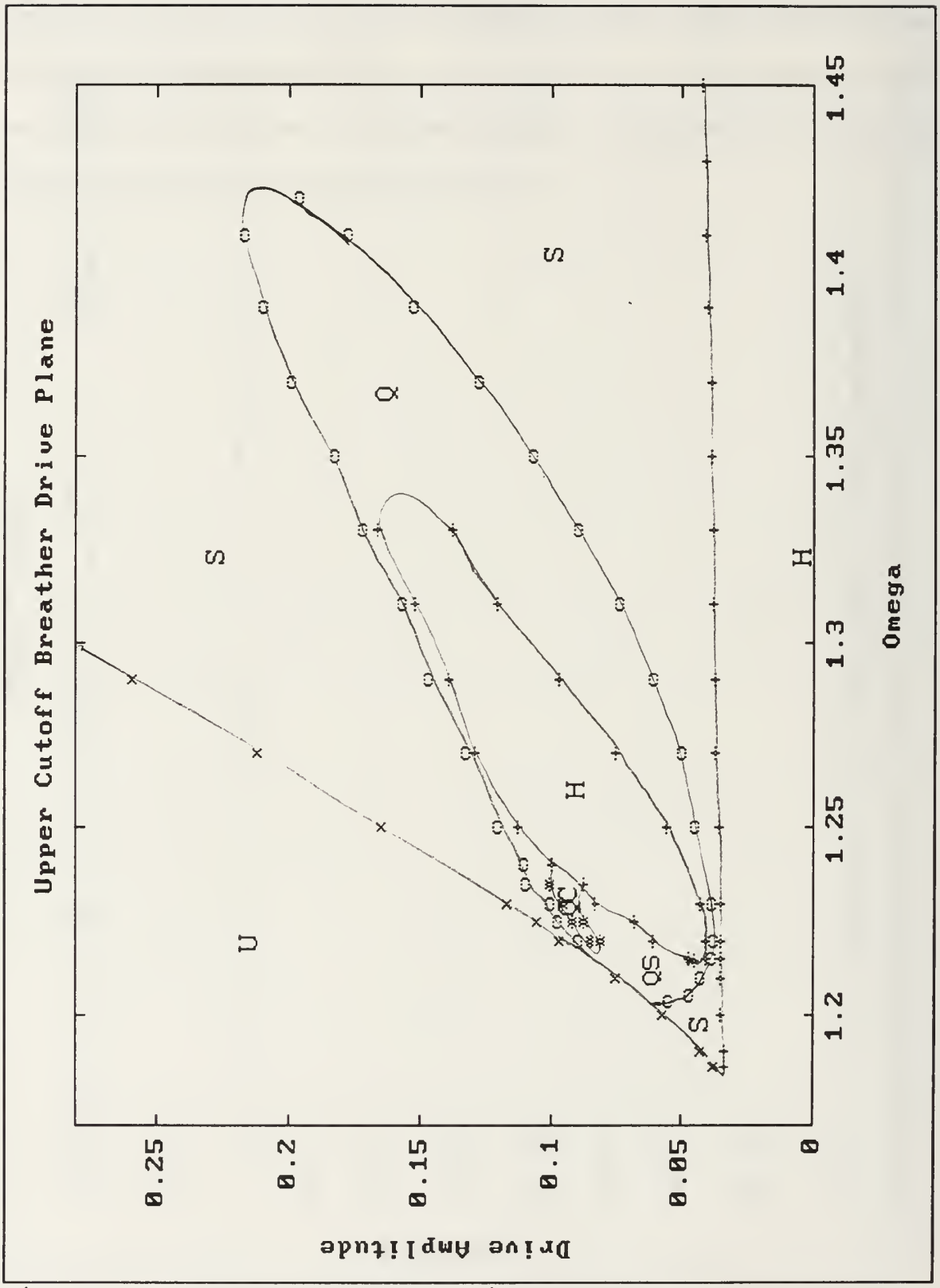

Figure III.C.2 Upper Cutoff Drive Plane

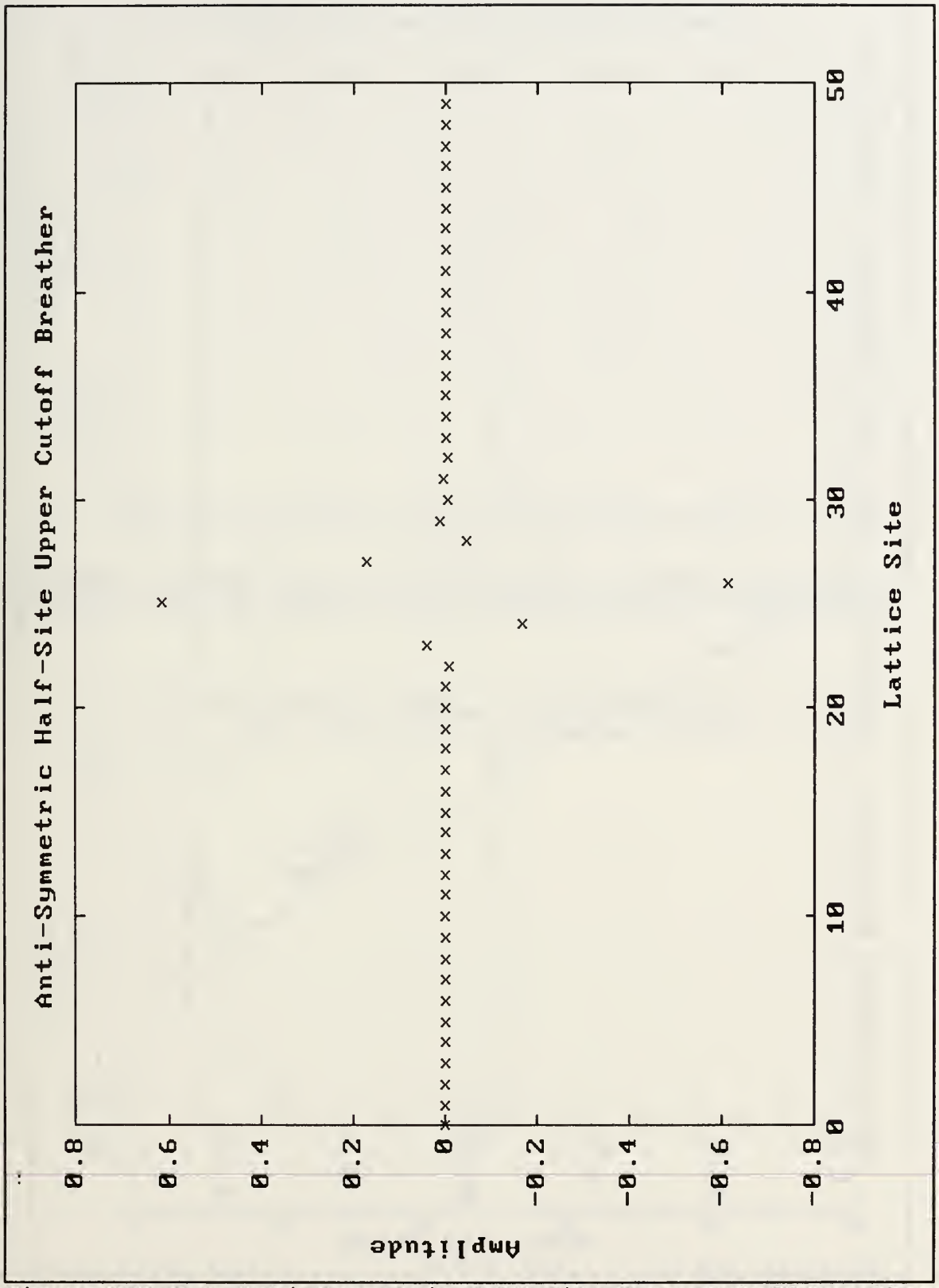

Figure III.C.3 Anti-Symmetric Half-Site Upper Cutoff Breather

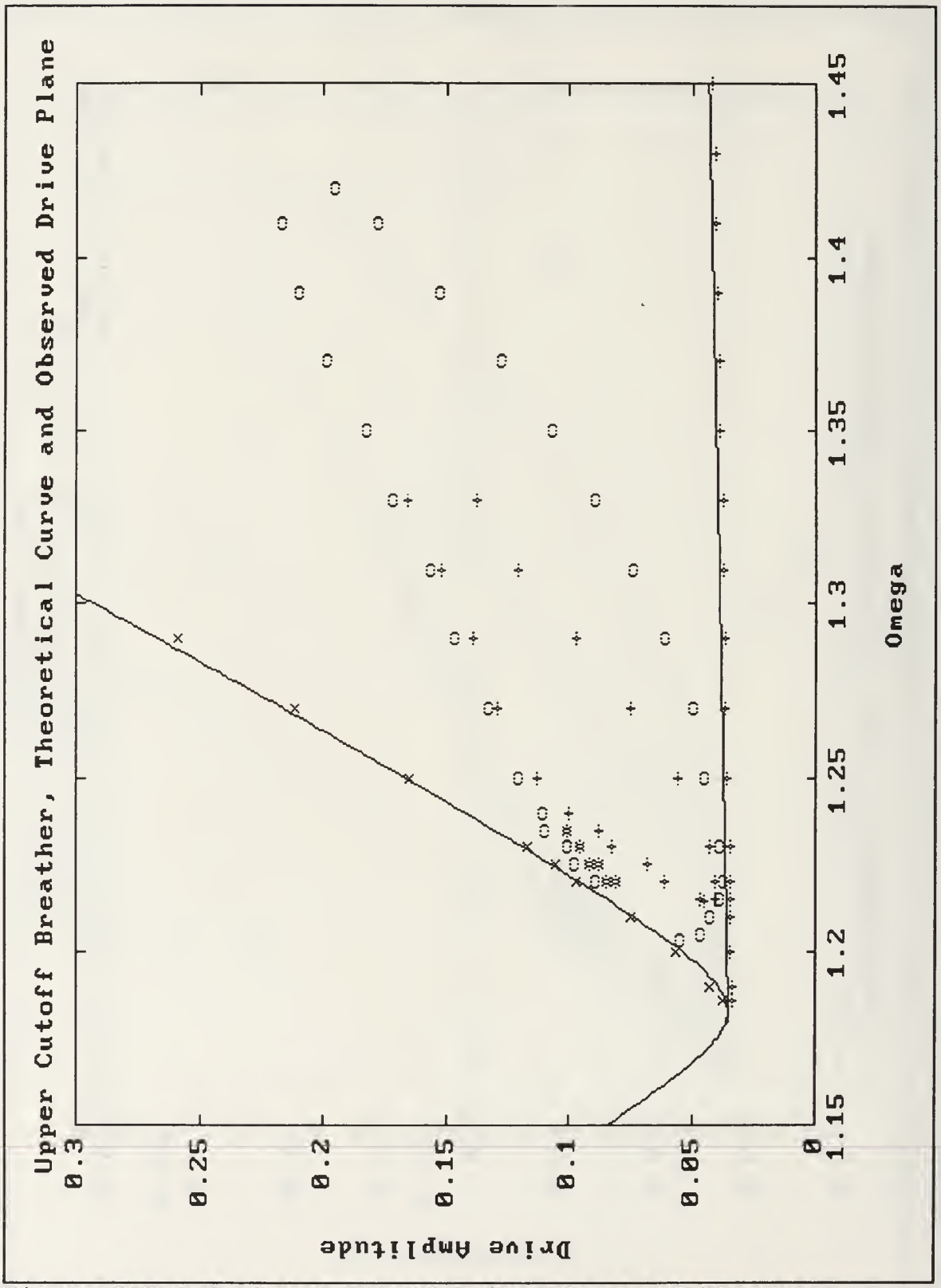

Figure III.C.4 Upper Cutoff Drive Plane and Theory

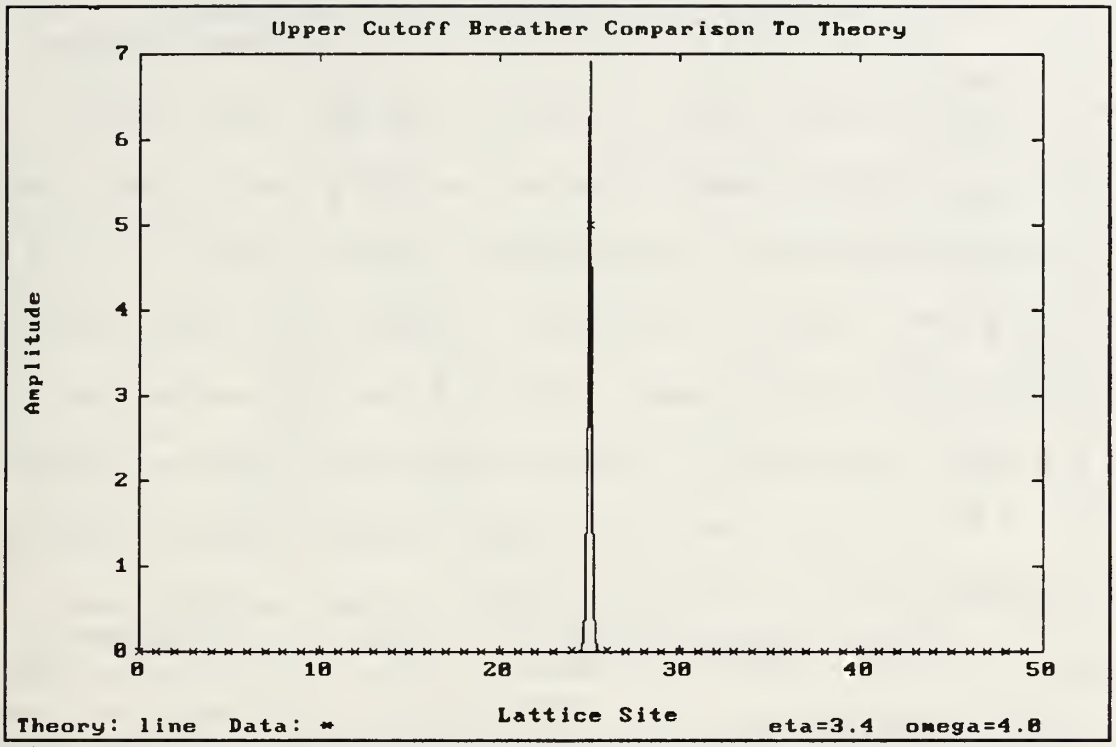

Figure III.C.5 Upper Cutoff Breather Theory Comparison

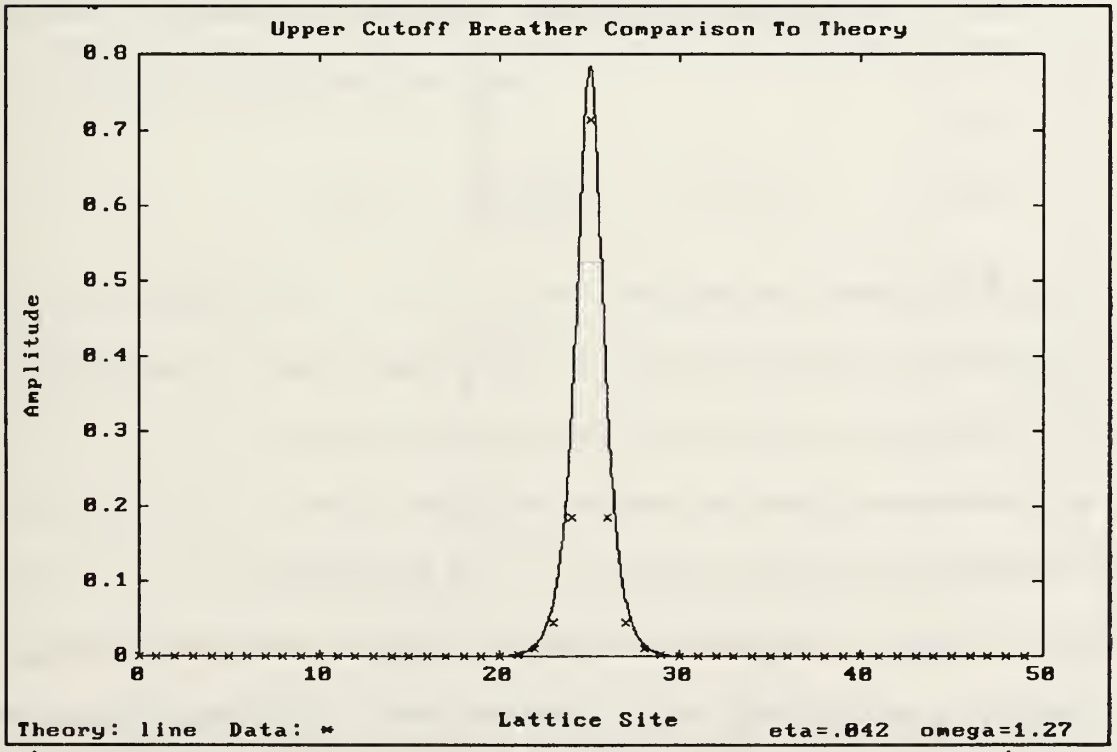

Figure III.C.6 Upper Cutoff Breather Theory Comparison

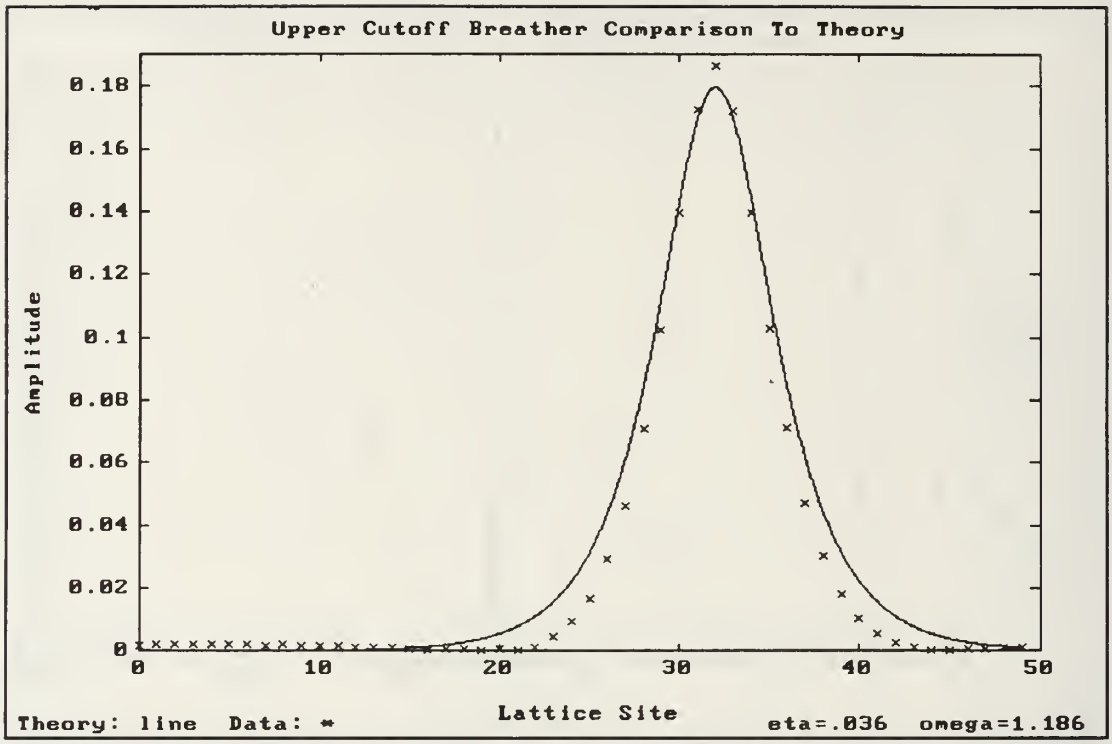

Figure III.C.7 Upper Cutoff Breather Theory Comparison

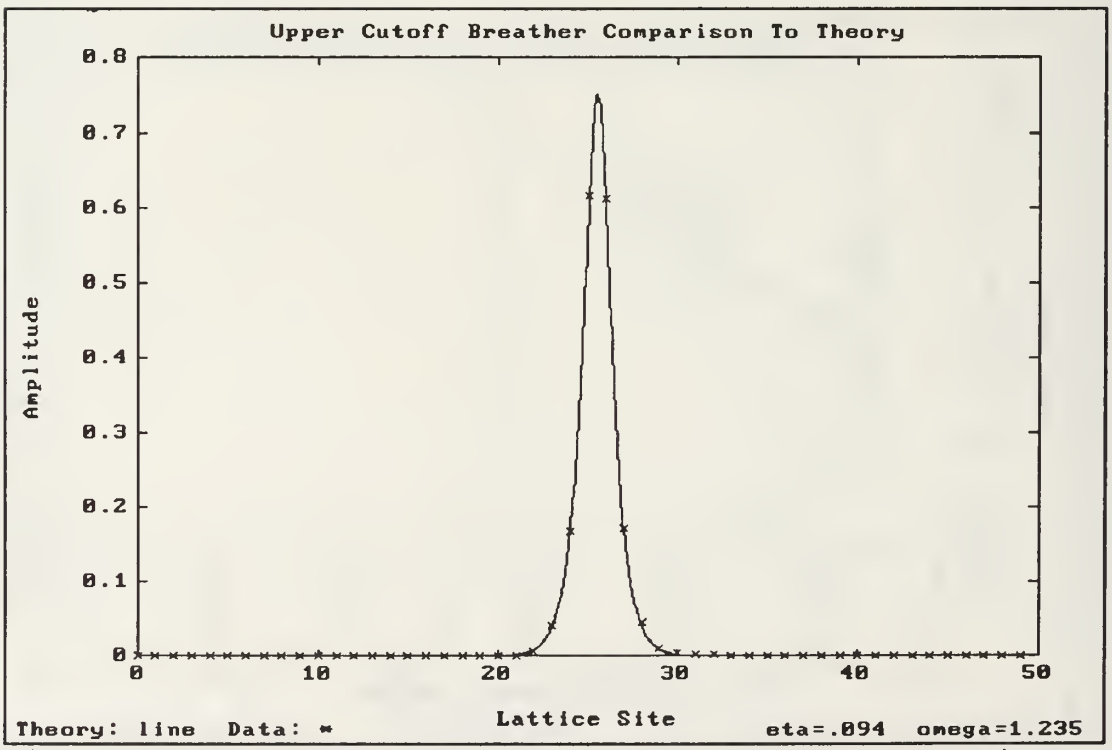

Figure III.C.8 Upper Cutoff Breather Theory Comparison

D. SOLITON SHEDDING

The initial states for both upper and lower cutoff mode response investigations were evanescent states in a lattice of 100 sites. One end of the lattice was driven and the other end was given a free boundary condition. The large number of sites minimized reflection while maintaining the speed of the simulation. The parameters for the upper cutoff were $\gamma=.75$, $\beta=.03$, while the lower cutoff were $\gamma=.50$, $\beta=.03$. Snapshots in time of the on-screen motion of the lattice were obtained to help describe the motion. In these "snapshot" figures, each dot is an individual lattice site and the site furthest to the left is the driven end site.

Soliton shedding was observed in both the upper and lower cutoff, end driven lattices. The shedding was similar for both modes, with differences in the actual method of energy transfer from the driven site. The upper cutoff shedding was produced by the cyclic relaxation in amplitude of the driven end site. The lower cutoff shedding was also generated by end site relaxation but transferred energy through a "pivot" point before the soliton was created.

An artificial potential well of the form:

$$f(x) = \frac{-x}{\sqrt{1 + x^2}} \qquad III.D.1$$

was used to replace $f(x)= -x + x^3$ in the lower cutoff program to permit amplitudes of the magnitude necessary to observe shedding in the softening mode. Otherwise, the oscillators would "go over the top." The upper threshold was marked by a distinct motion change and soliton shedding (Fig. III.D.1). The shed solitons were very small and strongly evanescent from $\omega=.95$ down to $\omega=.6$. The soliton shedding at $\omega=.6$ was distinct and the structure easily visible until 20-30 sites from the driven end. The shedding results were similar for $\omega=.5$ with solitons visible until approximately site 35. A third harmonic component appeared at low drive amplitudes

($\eta = .144$) and continued until the shedding threshold. The harmonic was identified by taking the FFT of a time series from the driven site (Fig. III.D.2). The visible result of the harmonic was an evanescent ripple wave (Fig. III.D.3).

A fifth harmonic was identified for $\omega = .3$ (Fig. III.D.4) and it also disappeared at the shedding threshold. The strength of the harmonic appeared to vary slowly with amplitude but over a fairly large range. The driven end site became quasiperiodic from $\eta = .415$ to 1.42 and from 2.14 to the shedding threshold for $\omega = .2$. Small wave packet ejection was observed from $\eta = 2.00$ to 2.59. Upon reaching the shedding threshold, the end site relaxed and a large soliton was shed (Fig. III.D.5). The soliton would propagate 6-8 sites, then appear to shed a much smaller soliton. Both the original and shed soliton would then damp out quickly (Fig. III.D.6). A seventh harmonic appeared for $\omega = .2$ but was not visually identifiable. Odd harmonics up to the 11th were identified for $\omega = .1$ (Fig. III.D.7-8) and visually affected the lattice motion. The end site quasiperiodicity started at $\eta = 2.25$ and the evanescent wave packets started at $\eta = 1.35$. The shedding threshold was not reached by $\eta = 6.0$.

The dynamic threshold for the upper cutoff lattice exhibits a consistent trend even though the motion of the resultant state changes significantly between $\eta = 2.32$ and 2.33 (Fig. III.D.9). The basic motion changes from soliton shedding to a non-shedding evanescent state. Shedding for $\omega = 2.025$ occurs very slowly and the soliton propagates 30-40 lattice sites and then appears to stop (Fig. III.D.10). Apparently the nonlinearity lowers the frequency enough that linear wave propagation can occur inside the structure and it appears to "ring" as it decays (Fig. III.D.11). The shedding occurs slowly for $\omega = 2.05$ and the soliton is large but jumbled (Fig. III.D.12) and dies out quickly. The size of the shed solitons is cyclic for $\omega = 2.1$-2.3 and the structure is very diffuse (Fig. III.D.13). Several small solitons are ejected followed by one large one and the relaxation of the end site is very distinct. Intermittent shedding occurs for $\omega = 2.31$ and the first

8-9 sites move together (Fig. III.D.14). The transition point from shedding to nonshedding seems to be between $\omega=2.32$ and 2.33. The motion for $\omega=2.32$ is varied in the number of sites in tandem, the size of the shed solitons and the magnitude of the end site relaxation. The shedding is transitional for $\omega=2.33$ and the motion settles into a complex evanescent state. The first 6-9 sites move in tandem and the last site is the first of 2-3 evanescent sites. The motion for $\omega=2.35$-2.6 is similar but only 2-3 sites are in tandem. The vertical boundary between shedding and non-shedding states was confirmed by obtaining high drive level evanescent states and crossing the boundary into the shedding region. Specifically, the states $\omega=2.35$ $\eta=5.05$ and $\omega=2.5$ $\eta=8.0$ still exhibited the same lattice motion initiatated at the transition boundary. The states were then decremented in frequency to cross into the shedding region. The motion of both states slowly evolved to the previously described lattice motion for the final frequency.

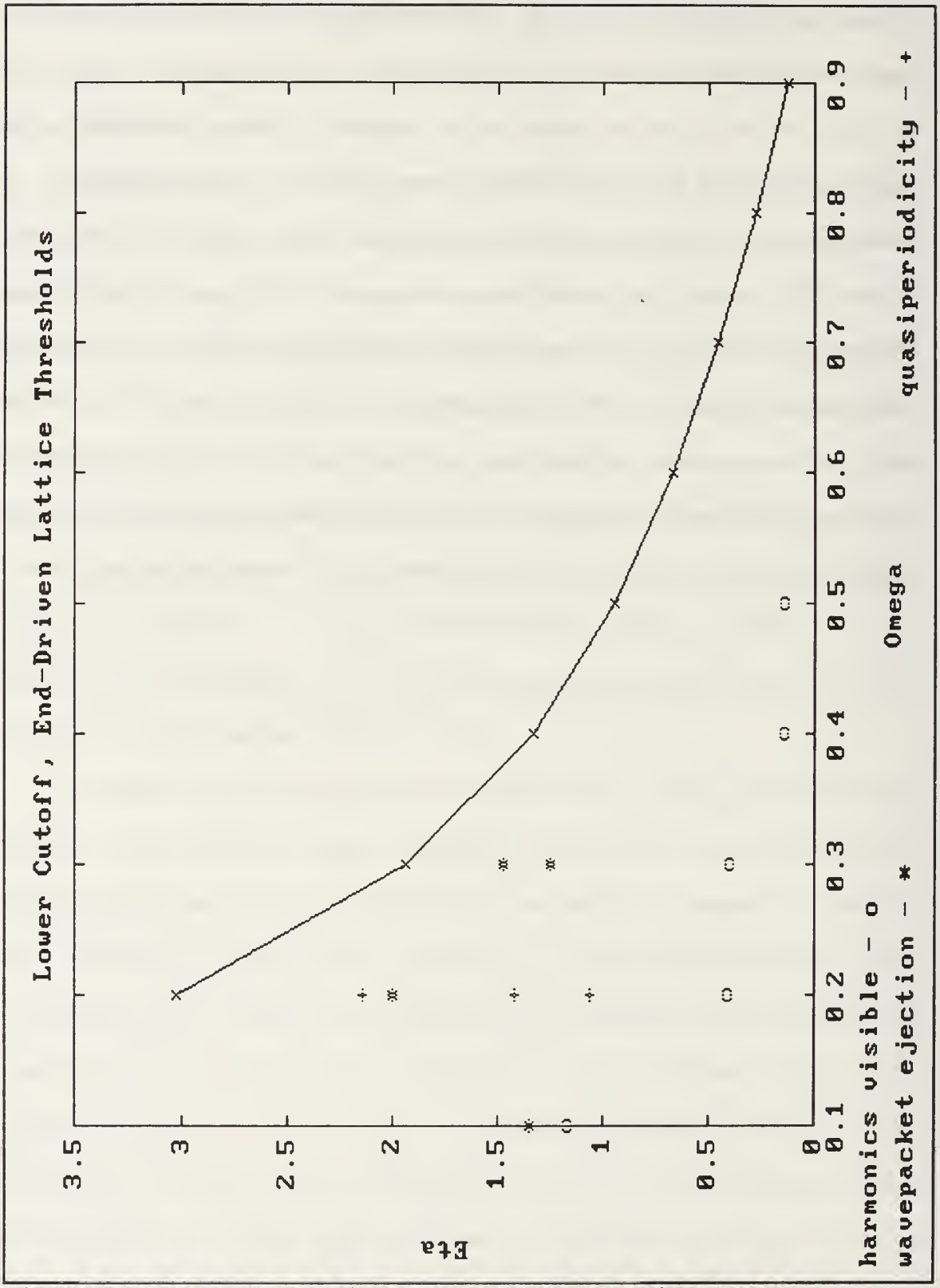

Figure III.D.1 End-Driven Lower Cutoff Drive Plane

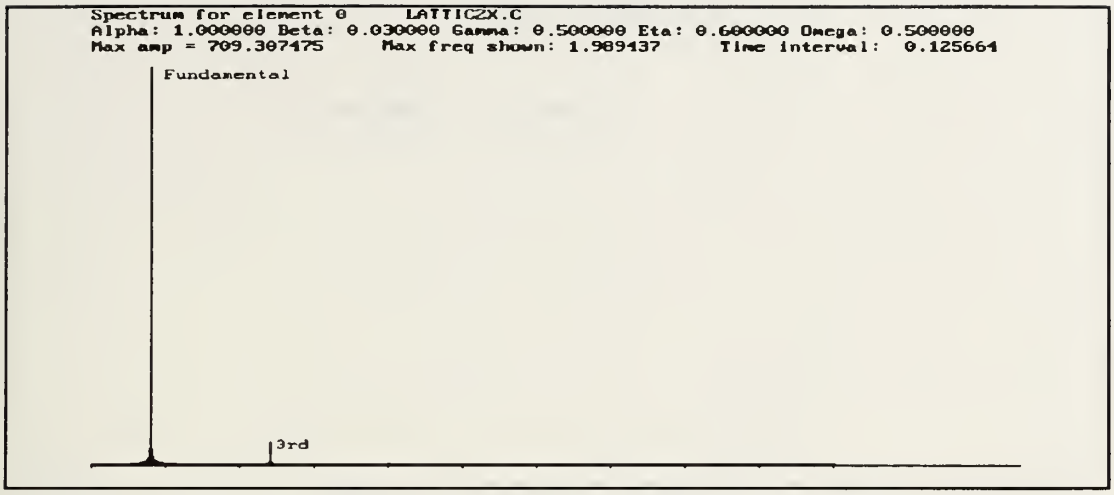

Figure III.D.2 Lower Cutoff FFT Spectrum ω=.5

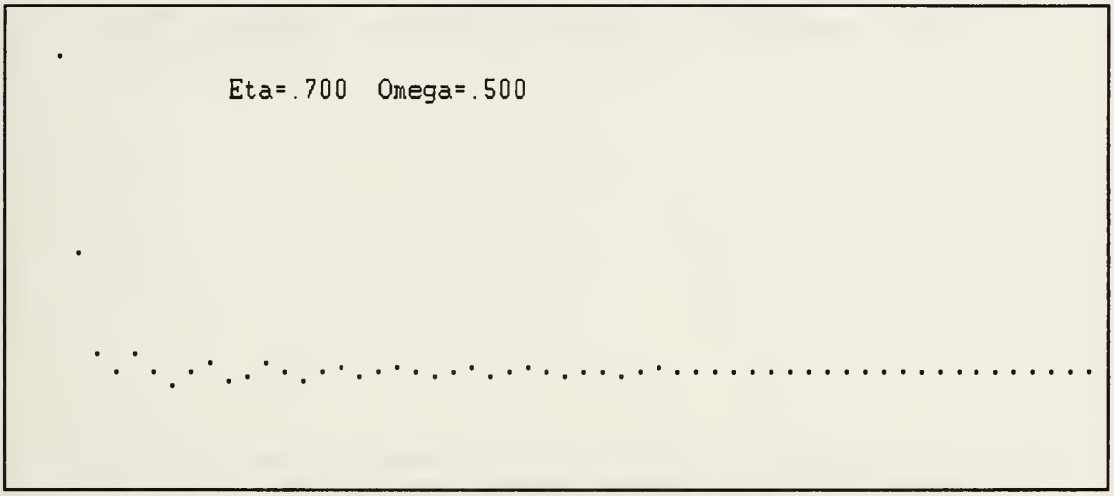

Figure III.D.3 Lower Cutoff Harmonic Ripple ω=.5

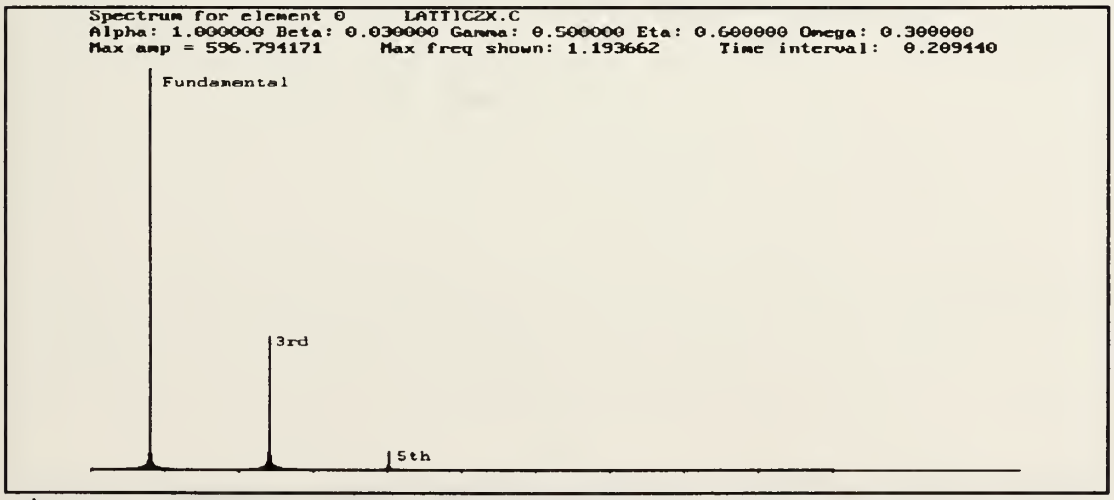

Figure III.D.4 Lower Cutoff FFT Spectrum ω=.3

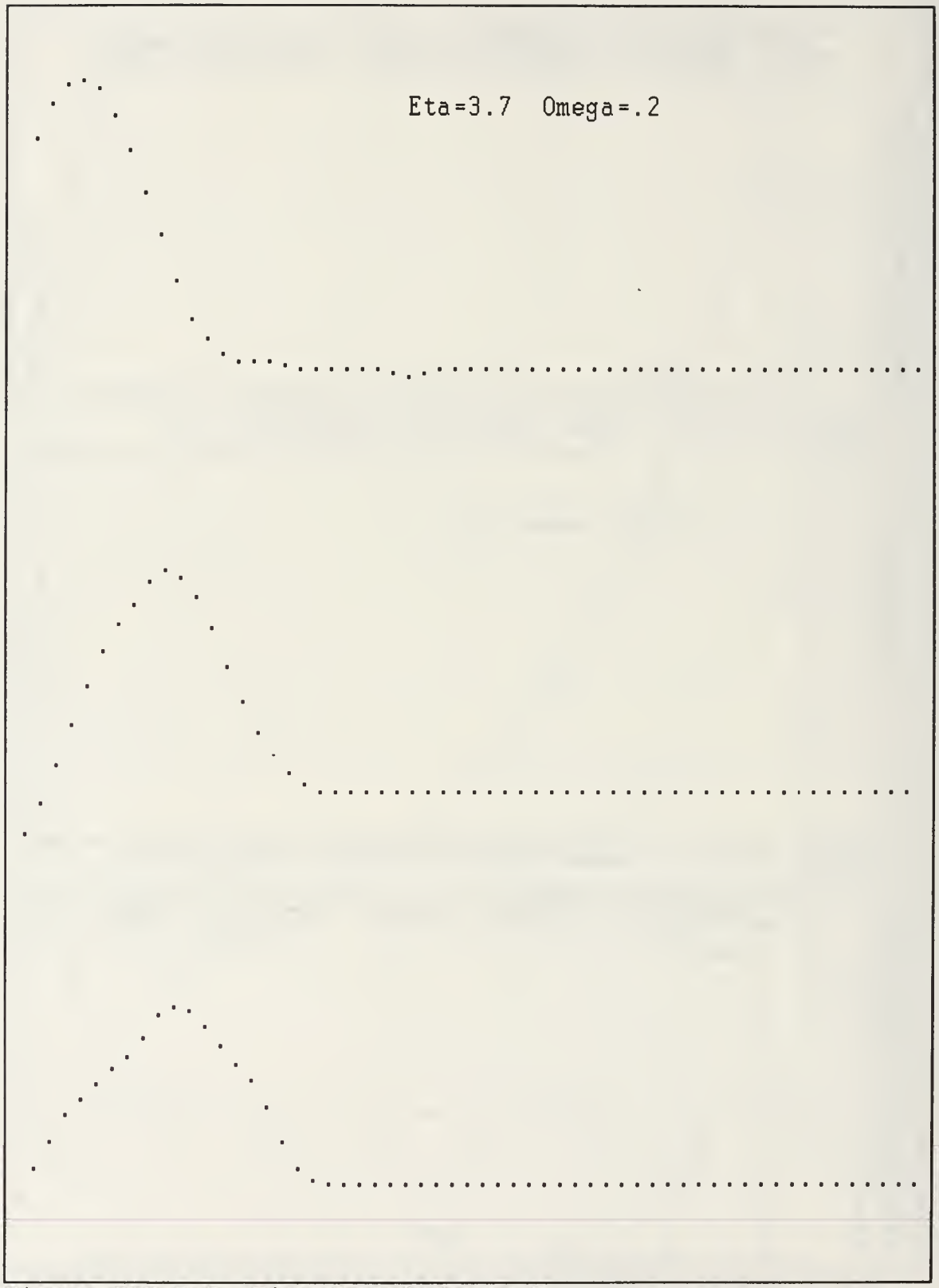

Figure III.D.5 Time Snapshots ω=.2

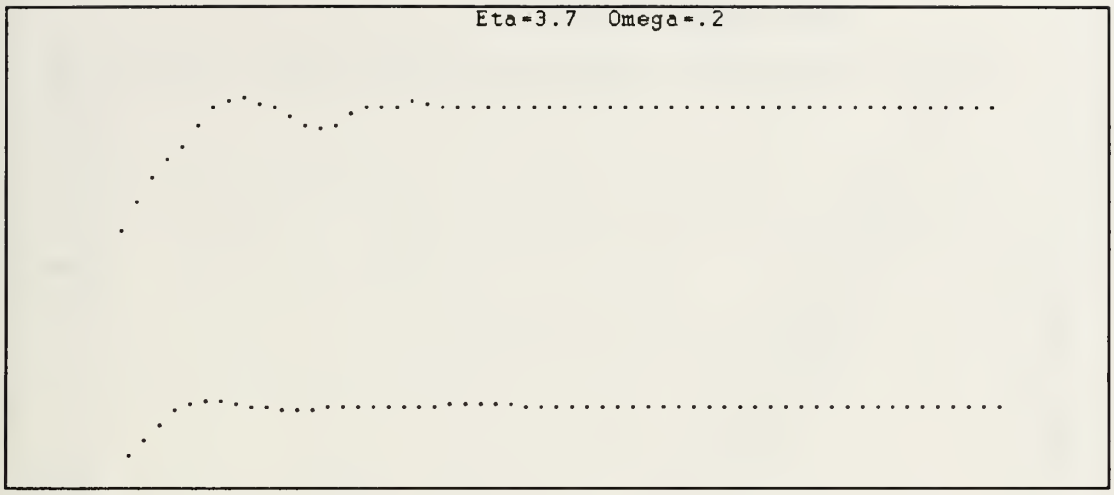

Figure III.D.6 Time Snapshots ω=.2

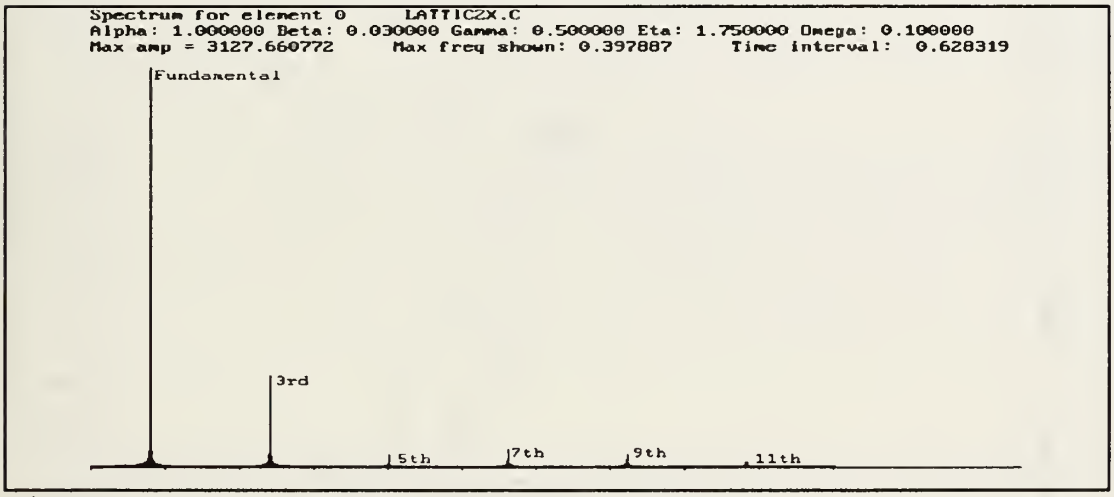

Figure III.D.7 Lower Cutoff FFT Spectrum ω=.1

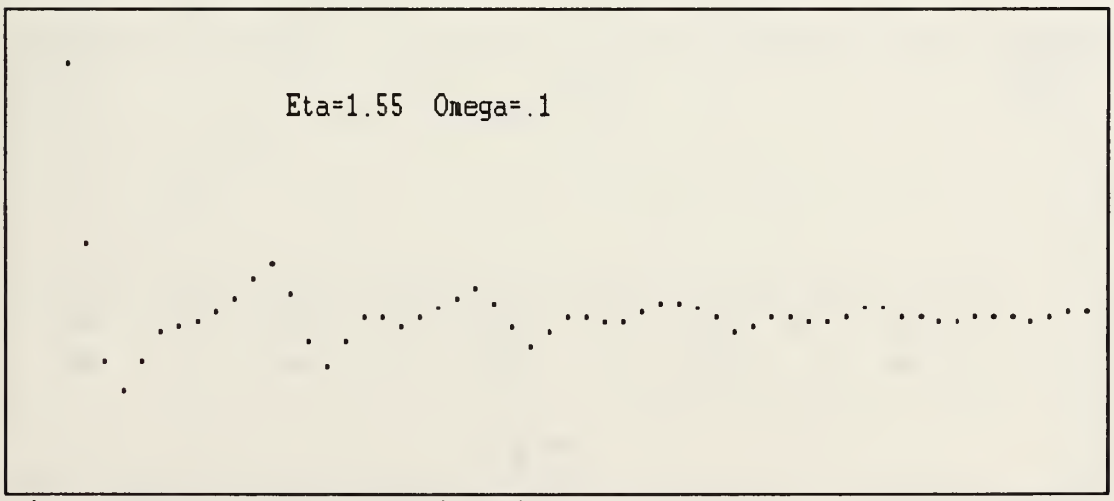

Figure III.D.8 Harmonic Ripples ω=.1

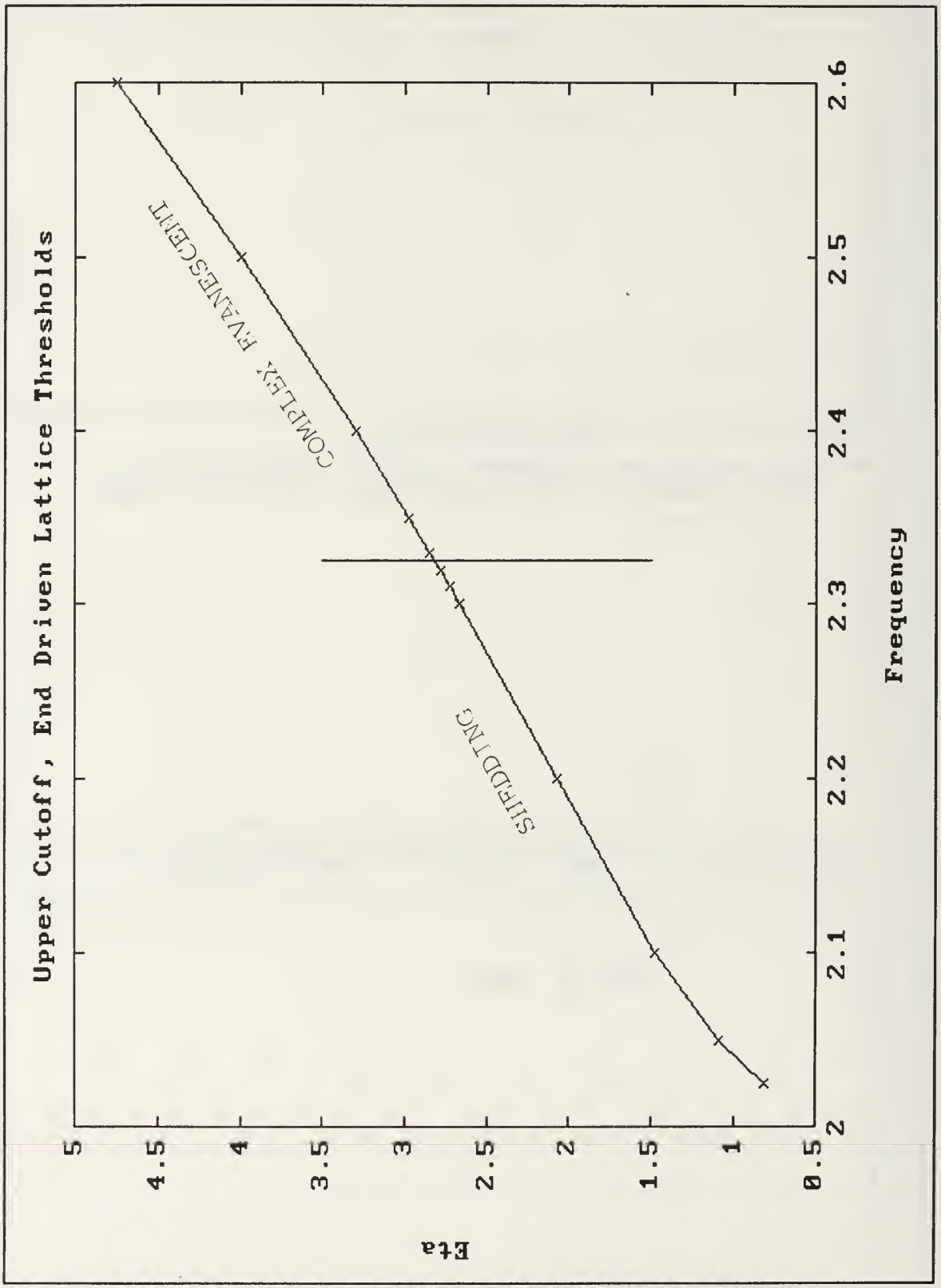

Figure III.D.9 End-Driven Upper Cutoff Drive Plane

Figure III.D.10 Time Snapshots ω=2.025

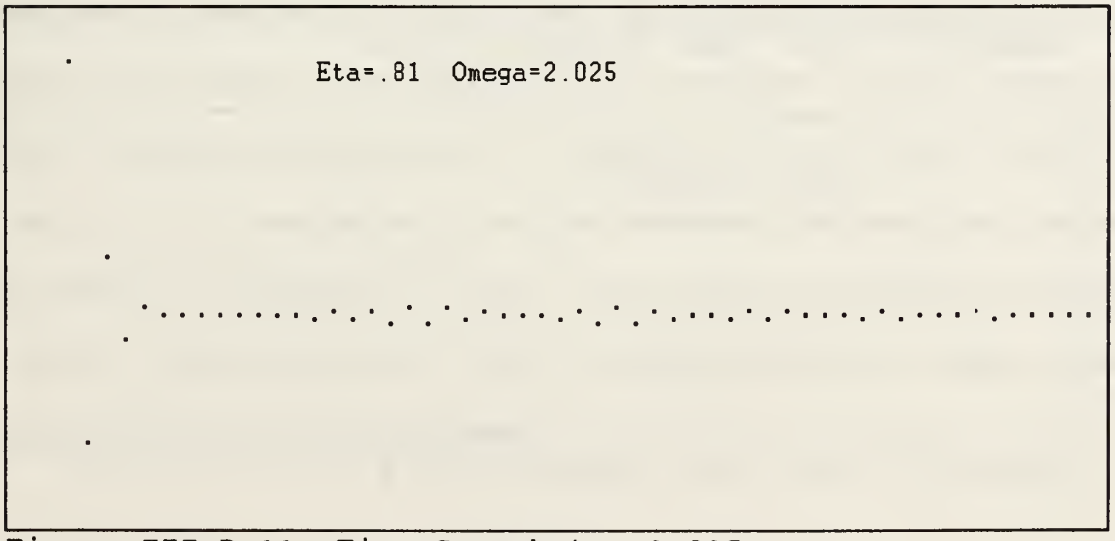

Figure III.D.11 Time Snapshot ω=2.025

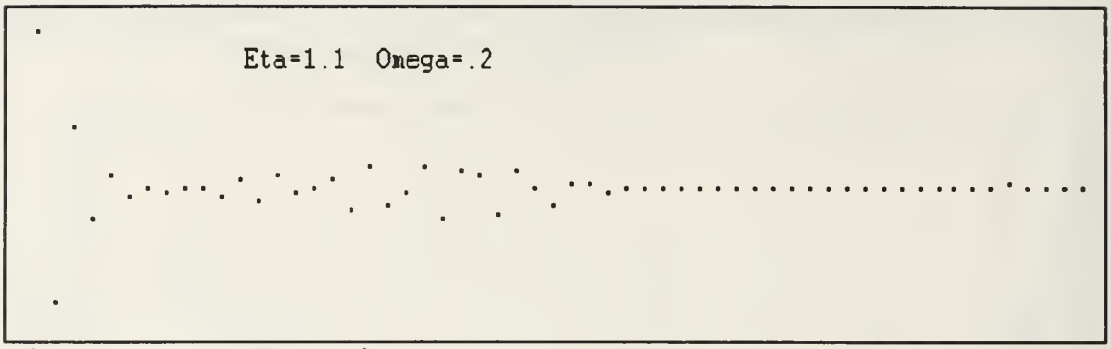

Figure III.D.12 Time Snapshot ω=2.05

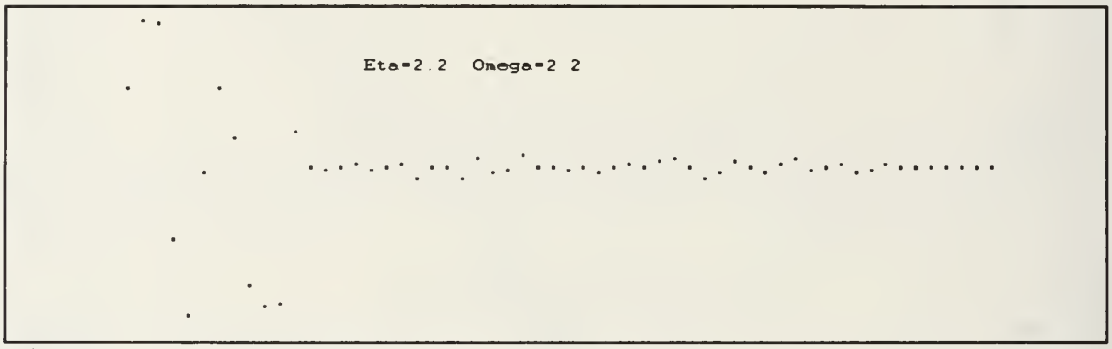

Figure III.D.13 Time Snapshot ω=2.2

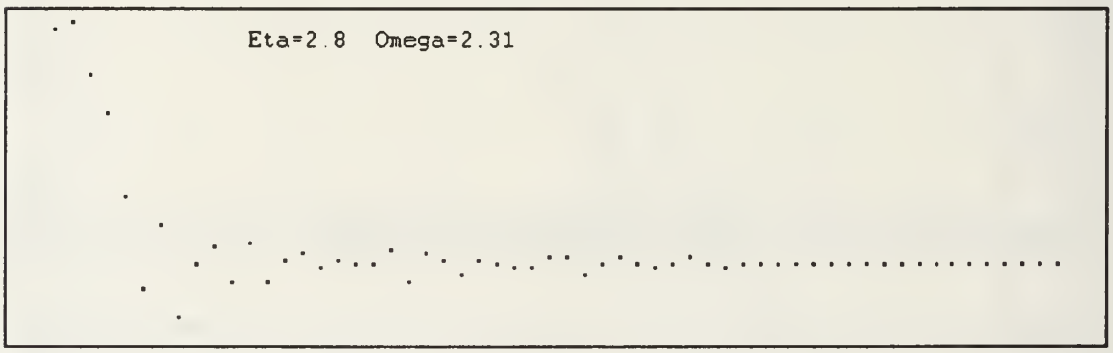

Figure III.D.14 Time Snapshot ω=2.31

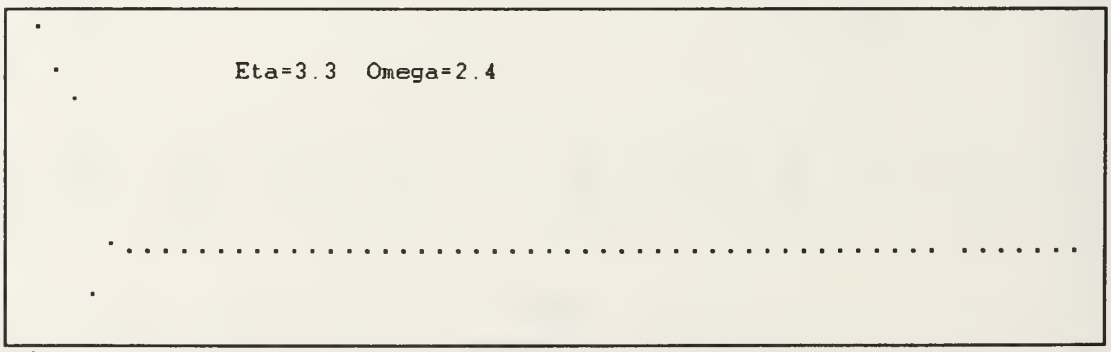

Figure III.D.15 Time Snapshot ω=2.4

IV. SUMMARY AND CONCLUSIONS

It has been shown numerically that nonpropagating steady state breather solitons can occur in one-dimensional lattices of coupled nonlinear oscillators that are damped and parametrically driven. The simplicity of the model suggests that these states can exist in a variety of systems. Amplitude data agree well with a nonlinear Schrödinger theory at low amplitudes, but disagree substantially at higher amplitudes. The theory assumes that the amplitudes are weakly nonlinear and slowly varying in space. The continued existence of breathers at amplitudes where the theory breaks down is evidence of the robustness of these states. Dissipation and drive stabilize the breathers, although this is not yet understood.

Breathers exist for a range of drive parameters (amplitude and frequency), and these regions have been mapped for both lower and upper cutoff breathers. Each region substantially disagreed with the theoretical prediction and, moreover, the natures of the two numerical regions were strongly dissimilar. The lower cutoff breathers are found to have a low-amplitude instability that is not predicted by the theory. The drive parameter region of the upper cutoff breathers contains a relatively large "finger" in which quasiperiodicity occurs, and a relatively large "island" in which the motion decays to rest. The corresponding instabilities are not yet understood, but it appears that the quasiperiodicity is a consequence of soliton shedding even though the shedding is not apparent except in a small region of the drive plane. This is based on the observation of strong quasiperiodic amplitude modulation during shedding and relatively weak modulation when shedding is not visible. Clearly visible soliton shedding is not expected to occur for a global parametric drive because a propagating breather has a spatially varying phase, whereas the drive is spatially constant.

For an end driven lattice, soliton shedding is dramatically visible. There is a drive amplitude threshold for the shedding to occur, and the value decreases as the linear frequency is approached. The phenomenon, although not yet understood, is expected to occur in any nonlinear wave system that possesses breathers. A possible application is the regeneration of fiber optic solitons which attenuate with distance.

APPENDIX

A. PROGRAM COMMANDS

1. **a** - Manual frequency increment, adjustable with A.
2. **A** - Manual frequency increment adjustment, can be positive or negative.
3. **d** - Coarse damping adjustment (decreasing), nominally .01, can be changed in variable declaration section of source code (the variable is Beta_increment).
4. **D** - Coarse frequency adjustment (decreasing), nominally .001, can be changed in variable declaration section of source code (Omega_increment).
5. **Ctrl D** - Coarse drive amplitude adjustment (decreasing), nominally .001, can be changed in variable declaration section of source code (Eta_increment).
6. **e** - Fine damping adjustment (decreasing), 1/10 of coarse damping increment, can be changed in the main() section of the source code under the appropriate key hit section.
7. **E** - Fine frequency adjustment (decreasing), 1/10 of coarse frequency increment, can be changed in the main() part of the source code under the appropriate key hit section.
8. **Ctrl E** - Fine drive amplitude adjustment (decreasing), 1/10 of coarse drive amplitude increment, can be changed in the main() section of the source code under the appropriate key hit section.
9. **Ctrl F** - Time series record of one lattice site, prompts for a file name to store the series. This function works only in Text Mode (Ctrl T). The number and

periodicity of the samples are controlled in the eqn_motion() function section of the source code. Set for 80 samples, taken at the rate of one per time increment.

10. **G** - Coupling adjustment (gamma), displays current value asks for new value.
11. **Ctrl G** - Graphics mode, displays real time lattice motion. Also refreshes the screen without interfering with the lattice motion. Automatically invoked when the program is started or restarted.
12. **Ctrl H** - Total lattice energy monitor, can only be used in Text Mode.
13. **i** - Zoom in. The amplification of the screen is 2^n for n button pushes. The amplification is vertical only.
14. **Ctrl I** - Increases the strobing frequency of the phaseplot option, effectively increasing the sampling rate.
15. **Alt K** - Coupling modulation, Options are random, gradient or sine wave.
16. **Ctrl K** - Amplitude kick, specified lattice site by the desired amount. Amplitude and lattice site input from the keyboard.
17. **Ctrl L** - Pins specified lattice site. Acts like a toggle switch, reselection unpins the site.
18. **n** - Random perturbation of the lattice. Perturbs all sites with a random coordinate change of $\leq \pm 3\%$ of the maximum amplitude in the lattice.
19. **o** - Zoom out. The reduction of the screen is 2^n for n button pushes. The reduction is vertical only.
20. **Ctrl O** - Decreases the strobe frequency of the phaseplot option, effectively decreasing the sampling rate.

21. **p** - Pause program. Freezes execution of the program. Alternate method of obtaining a time series file.

22. **P** - Phaseplot mode. Selects up to five lattice site for phase space monitoring.

23. **Ctrl Q** - Exits the program.

24. **r** - User initiated state save. The lattice must be stable to within 1% (blue color) or the option will freeze the program until the 1% criterion is met. The option saves the current lattice modulation at the upper turning point of the lattice site being monitored.

25. **Ctrl R** - Restarts the program without going back out to DOS. Goes to the same screen that was displayed when the program was first started up.

26. **s** - Program freeze, captures lattice modulation when pushed. Any key will continue the program but the lattice will be at rest.

27. **S** - Monitors the stability of the specified lattice site. Works like a toggle switch. The lattice will change colors when the monitored site is within 1% of its average amplitude for the last twenty upper turning points.

28. **Ctrl S** - Turns off the drive.

29. **Ctrl T** - Text Mode. Prints system parameters on screen. Total lattice energy can be monitored in this mode.

30. **u** - Coarse damping adjustment (increasing), nominally .01, can be changed in variable declaration section of source code (variable is Beta_increment).

31. **Ctrl U** - Coarse drive amplitude adjustment (increasing), nominally .001, can be changed in variable declaration section of source code (Eta_increment).

32. **U** - Coarse Frequency adjustment (increasing), nominally .001, can be changed in variable declaration section of source code (Omega_increment).

33. **v** - Fine damping adjustment (increasing), 1/10 of coarse damping increment, can be changed in the main() section of the source code under the appropriate key hit section.

34. **Ctrl V** - Fine drive amplitude adjustment (increasing), 1/10 of coarse frequency increment, can be changed in the main() section of the source code under the appropriate key hit section.

35. **V** - Fine drive amplitude adjustment (increasing), 1/10 of coarse increment, can be changed in the main() part of the source code under the appropriate key hit section.

36. **w** - Displays peak lattice amplitude, works in Graphics Mode only.

37. **Ctrl W** - Turns on waterfall type display. Trends are easily noticed on this type of display.

38. **Ctrl X** - Selects lattice site to monitor for spectrum analysis.

39. **y** - Dump spectrum generated by Ctrl X.

40. **z** - Sets damping to zero.

41. **Z** - Sets frequency to zero.

42. **Ctrl Z** - Zeros the peak amplitude variable (variable automatically zeros when a parameter is changed).

B. PROGRAM CODE

The QuickC program codes for the aforementioned numerical implementations (Sect III) are listed below. The full program listing for Lower Cutoff (CCLATL.C) is included and the equation of motion for Upper Cutoff (CCLATU.C), Lower Cutoff end-driven (ENDL.C) and Upper Cutoff end-driven (ENDU.C).

1. Lower Cutoff, Global Drive

```
            /*    PROGRAM LATTICE (VGA)
VERSION 2.0  (QUICKC)
*/
#define  LAST_UPDATE 22 JUL 1991 BY CLEON WALDEN
/*
THIS PROGRAM SIMULATES A GENERAL LATTICE WITH EQUATIONS OF MOTION
THAT CAN BE SUBSTITUTED IN WHERE INDICATED.  VARIOUS INTERACTIVE
FEATURES ARE PROVIDED, WHICH ARE EXPLAINED IN THE PROGRAMMER'S
MANUAL FOR LATTICE. ENERGY MONITORING IS ADDED TO TEXT SCREEN.
A WATERFALL DISPLAY IS ADDED TO MONITOR SOLITON MOTION DUE TO
MEDIUM EFFECTS.
*/

#include "BIOS.H"
#include "graph.h"
#include "stdlib.h"
#include "CONIO.H"
#include "MATH.H"
#include "STDIO.H"
#include "time.h"
#include "direct.h"
#include "process.h"
#include "dos.h"

#define getrandom(min,max) ((rand() % (int)((max)-(min)))+(min)+1)
#define SQR(a) ((a)*(a))
#define CUB(a) ((a)*(a)*(a))
#define SWAP(a,b) tempr=(a);(a)=(b);(b)=tempr
#define DOFOR(i,to) for(i=0;i<to;i++)
#define PI 3.14159265359
#define SCREEN_CORRECTION_FACTOR 1.05
#define eta_increment 0.01
#define beta_increment 0.01
```

```
#define omega_increment 0.001
#define STABILITY_INCREMENT 0.01
#define DISPLAY_INCREMENT DINC
#define PHASE_INCREMENT PINC
#define MIDDLE_ELEMENT (no_pendulums/2)
#define text_flag flags[0]
#define graphics_flag flags[1]
#define file_dump_flag flags[2]
#define stability_flag flags[3]
#define peak_flag flags[4]
#define stable_flag flags[5]
#define phase_flag flags[6]
#define pause_flag flags[7]
#define pause_flag2 flags[8]
#define stop_flag flags[9]
#define spectrum_flag flags[10]
#define dump_spectrum_flag flags[11]
#define energy_flag flags[12]
#define waterfall_flag flags[13]
#define wait_flag flags[14]

/*    The dynamical variables are entered here.  */

double coordinate[150],momentum[150],old_coordinate[150],old_momentum[150],
 oldold_coordinate[150],oldold_momentum[150];

double acceleration,gamma[150],mean_gamma,beta,omega0[150],mean_omega0,
eta,omega,alpha,model_time,time_int,
max_amp,mode_amp,time_series_disp[8000],phase_elements[5],
peak_record[20],pinned_elements[150],DINC,PINC,period,
spectrum[4098],temp2[4096],energy,gradient,wat_inc,
peak_amp,amp_kick,freq_inc,delta;
int    no_pendulums,flags[15],counter1,phase_counter,phase_int,first_element,
chosen_element,stability_element,disp_color,colors[5],counter2,step_size,
spectrum_element,spectrum_counter,sample_counter,energy_counter,g,gst[8000],
  waterfall_counter,color_counter,number_clicks,p_flag,t_flag,top_flag;

main() {
FILE *fp;
int c,i,j,k,l,m,n,xx,a;
double modulation;
char ans[5],buff[50],buff2[50];
/*   void display_text(),stability_restart(),user_init(),display_graphics(),
process_pause(),monitor_stability(),start_file_dump(),
stop_file_dump(),pin_elements(),kick_elements(),stability_check(),
init_phasplot(),phasplot(),compute_spectrum(),plot_spectrum(); */
_clearscreen(_GCLEARSCREEN);
```

```
user_init();
_setvideomode(_MRES4COLOR);
_setcolor(1);
sprintf(buff,"Eta %.3lf  Omega %.3lf  Gamma %.2lf",eta,omega,mean_gamma);
  _registerfonts("TMSRB.FON");
  _moveto(10,10);
  _setfont("t'tms rmn'bn5");
  _outgtext(buff);
xx=0;
wat_inc=1;
disp_color=1;
stop_flag=0;
counter2=0;
energy_flag=0;
energy_counter=0;
color_counter=0;
while(stop_flag==0) {
energy=0;
if(pause_flag2==1) {
pause_flag2=0;
pause_flag=0;
process_pause();
}  /* if(pause...) */
if(kbhit()!=0) {          /* Check for keystroke from user */
c=getch();
if(c==17) {           /* ^Q      :  Quit program */
stop_flag=1;
}
else if(c==112) {     /* p       :  Pause program/dump state */
pause_flag=1;
}
else if(c==20)  {     /* ^T      :  Text Mode */
_clearscreen(_GCLEARSCREEN);
text_flag=1;
graphics_flag=0;
waterfall_flag=0;
wait_flag=0;
spectrum_flag=0;
display_text();
}
else if(c==7)  {      /* ^G      :  Graphics Mode */
screen_graph();
phase_flag=0;
wait_flag=0;
spectrum_flag=0;
waterfall_flag=0;
text_flag=0;
```

```
graphics_flag=1;
sprintf(buff,"Eta %.3lf  Omega %.3lf  Gamma %.2lf",eta,omega,mean_gamma);
_registerfonts("TMSRB.FON");
_moveto(10,10);
_setfont("t'tms rmn'bn5");
_outgtext(buff);
}
else if(c==8) {     /* ^H       : Monitor energy in text mode */
if(energy_flag==0) energy_flag=1;
else energy_flag=0;
}
else if(c==16) {    /* ^P       : Kick in parameter variations */
screen_text();
srand((unsigned)time(NULL));
printf("\nEnter 1 if you wish to vary coupling:  ");
scanf("%d",&j);
if(j==1) {
  printf("\nEnter 1 (random), 2 (gradient), or 3(sine) desired:  ");
  scanf("%d",&k);
  if(k==2) {
printf("\nEnter gradient (percentage over entire lattice):  ");
scanf("%lf",&gradient);
DOFOR(i,no_pendulums) {
gamma[i]=gamma[i]+mean_gamma*i*gradient/(100*no_pendulums);
} /* DOFOR */
  } /* if(j==2) */
  else if(k==3) {
 printf("\nEnter modulation amplitude (percentage): ");
 scanf("%lf",&gradient);
 printf("\nEnter integer number of wavelengths to use:  ");
 scanf("%d",&l);
 DOFOR(i,no_pendulums)  {
gamma[i]=gamma[i]+mean_gamma*gradient/100*(-cos(2*l*PI*i/
no_pendulums));
 }
  }
  else DOFOR(i,no_pendulums) {
j=rand();
gamma[i]=mean_gamma*(.95+.1*j/RAND_MAX);
  }
}
else {
  printf("\nVarying omegao...\n");
  printf("\nEnter 1 (random), 2 (gradient), or 3(sine) desired:  ");
  scanf("%d",&k);
  if(k==2) {
printf("\nEnter gradient (percentage over entire lattice):  ");
```

```
scanf("%lf",&gradient);
DOFOR(i,no_pendulums) {
  omega0[i]=omega0[i]+mean_omega0*i*gradient/(100*no_pendulums);
}
  }
  else if(k==3) {
printf("\nEnter modulation amplitude (percentage):  ");
scanf("%lf",&gradient);
printf("\nEnter integer number of wavelengths to use:  ");
scanf("%d",&l);
DOFOR(i,no_pendulums) {
omega0[i]=omega0[i]+mean_omega0*gradient/100*(-cos(2*l*PI*i/no_pendulums));
}
  }
else {
DOFOR(i,no_pendulums) {
j=rand();
printf("Random number is: %d  RAND_MAX is %d\n",j,RAND_MAX);
omega0[i]=mean_omega0*(.95+.1*j/RAND_MAX);
}
}
}
screen_graph();
}
else if(c==23)  {     /* ^W        :  Waterfall Display Mode */
if(waterfall_flag==0) {
phase_flag=0;
graphics_flag=0;
text_flag=0;
spectrum_flag=0;
waterfall_flag=1;
init_waterfall();
}
else {
if(wait_flag==1) {
wait_flag=0;
init_waterfall();
}
else waterfall_flag=0;
}
}
else if(c==80)  {     /* P         :  Phase Plot Mode */
if(phase_flag==0) init_phasplot();
else {
phase_flag=0;
spectrum_flag=0;
graphics_flag=1;
```

```
waterfall_flag=0;
wait_flag=0;
screen_graph();
} /* else */
}
else if(c==6) {       /* ^F       : Time series to file */
if(file_dump_flag==0) start_file_dump();
else {
stop_file_dump();
file_dump_flag=0;
}
}
else if(c==47) {      /* G        : Change gamma */
 screen_text();
 printf("\nEnter new gamma, gamma is %lf",mean_gamma);
 scanf("%lf",&mean_gamma);
 DOFOR(i,no_pendulums) {
gamma[i]=mean_gamma;
 }
 screen_graph();
 stability_restart();
}
else if(c==119) {     /* w        : Peak amplitude */
 _moveto(210,10);
 _setfont("t'tms rmn'bn5");
  sprintf(buff2,"Amp  %.4lf",peak_amp);
 _outgtext(buff2);
}
else if(c==26) {      /* ^Z       : Reset peak_amp variable */
 peak_amp=0.0;
}


else if(c==24) {      /* ^X       : Calculate spectrum */
spectrum_flag=1;
screen_text();
printf("\nEnter number of element to be analyzed:  ");
scanf("%d",&spectrum_element);
screen_graph();
}
else if(c==21) {      /* ^U       : Increase drive (coarse) */
                  save_state();
                  eta=eta+eta_increment;
stability_restart();
}
else if(c==4) {       /* ^D       : Decrease drive  (coarse) */
                  save_state();
```

```
                eta=eta-eta_increment;
stability_restart();
}
else if(c==22) {      /* ^V       : Increase drive  (fine)  */
                save_state();
                eta=eta+eta_increment/10;
stability_restart();
}
else if(c==5)  {      /* ^E       : Decrease drive   (fine) */
                save_state();
                eta=eta-eta_increment/10;
stability_restart();
}
else if(c==26) {      /* ^S       : Turn off drive */
                save_state();
eta=0;
                stability_restart();
}
else if(c==85) {      /* U        : Increase frequency (coarse)*/
                save_state();
time_wait();
time_int=time_int*200/period;
omega=omega+omega_increment;
period=2*PI/omega;
time_int=time_int*period/200;
stability_restart();
}
else if(c==68) {      /* D        : Decrease frequency (coarse)*/
                save_state();
time_wait();
time_int=time_int*200/period;
omega=omega-omega_increment;
period=2*PI/omega;
time_int=time_int*period/200;
stability_restart();
}
else if(c==86) {      /* V        : Increase frequency (fine)*/
                save_state();
time_wait();
time_int=time_int*200/period;
omega=omega+omega_increment/10;
period=2*PI/omega;
time_int=time_int*period/200;
stability_restart();
}
else if(c==69) {      /* E        : Decrease frequency (fine)*/
  save_state();
```

```
time_wait();
time_int=time_int*200/period;
omega=omega-omega_increment/10;
period=2*PI/omega;
time_int=time_int*period/200;
stability_restart();
}
        else if(c==90) {      /* Z       : Set frequency to zero */
                save_state();
omega=0;
stability_restart();
}
else if(c==65)  {     /* ^A       : Set manual frequency increment*/
  screen_text();
  printf("Manual frequency increment is %lf\n",freq_inc);
  printf("New increment:");
  scanf("%lf",&freq_inc);
  screen_graph();
}
else if(c==97)  {     /* a        : Manual Frequency increment*/
  save_state();
time_wait();
time_int=time_int*200/period;
omega=omega+freq_inc;
period=2*PI/omega;
time_int=time_int*period/200;
stability_restart();
}
else if(c==117) {     /* u        : Increase damping (coarse)*/
                save_state();
beta=beta+beta_increment;
stability_restart();
}
else if(c==100) {      /* d        : Decrease damping (coarse)*/
  save_state();
beta=beta-beta_increment;
stability_restart();
}
else if(c==118) {       /* v        : Increase damping (fine)*/
                save_state();
beta=beta+beta_increment/10;
stability_restart();
}
else if(c==121) {       /* y        : Dump spectrum  */
if(spectrum_flag==1) dump_spectrum_flag=1;
}
else if(c==101) {       /* e        : Decrease damping (fine)*/
```

```
                save_state();
beta=beta-beta_increment/10;
stability_restart();
}
else if(c==122) {       /* z       : Turn off damping */
                save_state();
beta=0;
stability_restart();
}
else if(c==12) {        /* ^L      : Pin elements */
pin_elements();
stability_restart();
}
else if(c==11) {        /* ^K      : Kick elements */
kick_elements();
stability_restart();
}
else if(c==105) {       /* i       : Zoom in  */
DINC=DINC/2;
PINC=PINC*2;
wat_inc=wat_inc*2;
}
else if(c==111) {       /* o       : Zoom out  */
DINC=DINC*2;
PINC=PINC/2;
wat_inc=wat_inc/2;
}
else if(c==110) {       /* n       : Perturb system */
/* Maximum +/-3% of peak value */
  linear_kick();
  screen_graph();
  stability_restart();
}
else if(c==15) {        /* ^O      : Decrease strobe freq */
phase_int=phase_int-1;
}
else if(c==9)  {        /* ^I      : Increase strobe freq */
phase_int=phase_int+1;
}
else if(c==18) {        /* ^R      : Restart program */
screen_text();
user_init();
sprintf(buff,"Eta %.3lf  Omega %.3lf  Gamma %.2lf",eta,omega,mean_gamma);
_registerfonts("TMSRB.FON");
_moveto(10,10);
_setfont("t'tms rmn'bn5");
_outgtext(buff);
```

```
if(phase_flag==0) _setvideomode(_MRES4COLOR);
else _setvideomode(_VRES16COLOR);
}
else if(c==115) {       /* s       : Stop everything as is */
DOFOR(i,no_pendulums) {
coordinate[i]=0;
momentum[i]=0;
} /* DOFOR */
stability_restart();
}
else if(c==83) {        /* S       : Monitor stability */
monitor_stability();
sprintf(buff,"Eta %.3lf  Omega %.3lf  Gamma %.2lf",eta,omega,mean_gamma);
_registerfonts("TMSRB.FON");
_moveto(10,10);
_setfont("t'tms rmn'bn5");
_outgtext(buff);
}
else if(c==114) { /*  r      : User initiated state save */
                save_stateu();
        }

}   /* if(kbhit...) */
if(wait_flag==0) {
eqn_motion();
} /* if(wait_flag */
} /* while(stop_flag... */
_setvideomode(_DEFAULTMODE);
printf("PROGRAM COMPLETE AT %lf",model_time);
}  /* MAIN() */

user_init() {
FILE *fq;
int c,i,j,k,l,m;
char answer[1],ans[1],answ[1],filename[30];
printf("GENERALIZED LATTICE MODEL PROGRAM W/ VARIABLE PARAMETERS\n");
printf("Version 2.1X486 (MS) \n");
printf("Last updated 22 JUL 1991\n");
printf("Variant notes: Default IC is AM\n");
printf("Omega0 is set equal to one for all cases!\n");
printf("Rotating phase plane is used.\n");
printf("Real time FFT function is added...\n");
printf("Energy monitoring available via ^H in text mode\n");
printf("Set top_flag to get output files for theory comparison.\n");
printf("Manual Frequency increment is set to 1/20 omega inc.\n");
/* Sets flag to output additional information for theory comparison.
Adds the information to the end of the normal output file*/
```

```
printf("Set top_flag?  ");
scanf("%s",ans);
if((ans[0]==89)||(ans[0]==121)) {
top_flag=1;}
else top_flag=0;
printf("\nDo you want to use a file for initial conditions (Y/N)?  ");
scanf("%s",answer);
sample_counter=0;
if((answer[0]==89)||(answer[0]==121)) {
printf("\nOld file?");
scanf("%s",answ);
printf("\nEnter name of file to be read:  ");
scanf("%s",filename);
        if((fq=fopen(filename,"r"))!=NULL) {
mean_gamma=0;
fscanf(fq,"%d\n",&no_pendulums);
fscanf(fq,"%lf\n",&alpha);
fscanf(fq,"%lf\n",&beta);
fscanf(fq,"%lf\n",&eta);
fscanf(fq,"%lf\n",&omega);
DOFOR(i,no_pendulums) {
fscanf(fq,"%lf  %lf  %lf  %lf\n",
&omega0[i],&gamma[i],&coordinate[i],&momentum[i]);
mean_gamma=mean_gamma+gamma[i];
}
mean_gamma=mean_gamma/no_pendulums;
fclose(fq);
}
else printf("Can't open file requested.");
} /* if((ans... */
else {
printf("\nEnter number of pendulums to use:  ");
scanf("%d",&no_pendulums);
printf("\nEnter mode amplitude:  ");
scanf("%lf",&mode_amp);
printf("\nEnter modulation amplitude:  ");
scanf("%lf",&max_amp);
printf("\nEnter nonlinear coefficient alpha (+/- 1 ONLY):  ");
scanf("%lf",&alpha);
DOFOR(k,no_pendulums) {
if(alpha<0) coordinate[k]=pow((-1),k)*(mode_amp
+max_amp*sin(2*PI*k/no_pendulums));
else coordinate[k]=mode_amp+max_amp*sin(2*PI*k/no_pendulums);
momentum[k]=0;
pinned_elements[k]=0;
}  /* DOFOR */
printf("\nEnter coupling coefficient gamma:  ");
```

```
scanf("%lf",&mean_gamma);
DOFOR(i,no_pendulums) {
gamma[i]=mean_gamma;
omega0[i]=1;
}
printf("\nEnter drive amplitude eta:  ");
scanf("%lf",&eta);
printf("\nEnter drive frequency omega:  ");
scanf("%lf",&omega);
printf("\nEnter dissipation constant beta:  ");
scanf("%lf",&beta);
}  /* else */
printf("\nEnter number of steps per response cycle: ");
scanf("%d",&step_size);
if ((step_size/2) != 0) g=.5;
else g=0;
time_int=2*PI/(step_size*omega);
DOFOR(i,15) flags[i]=0;
DOFOR(i,4096) spectrum[i]=0;
DOFOR(i,150) {
oldold_coordinate[i]=0.0;
old_coordinate[i]=0.0;
}
 screen_graph();
    number_clicks=0;
    mean_omega0=1.0;
peak_amp=0;
spectrum_counter=0;
/* Phase correction to start files in the proper phase*/
if((answ[0]==89)||(answ[0]==121)) delta=asin((omega*beta)/eta)/2;
else  delta=(asin((omega*beta)/eta)/2)-PI;
model_time=0.0;
freq_inc=omega_increment/20;
DINC=.02;       /* CHANGE THIS TO CHANGE SCALE OF DISPLAY */
PINC=2;         /* CHANGE THIS TO CHANGE SCALE OF PHASE PLOT */
} /* USER_INIT */


display_graphics() {
/* Displays the lattice sites.  The screen updates sequentially through the
points after each round of motion calculations are complete. */
int c,i,j,k,l,m,n;
   if(no_pendulums<=40) {
 first_element=0;
DOFOR(k,no_pendulums) {
n=0;
if((stability_element==k)&&(stability_flag==1)) n=1;
```

```
if((pinned_elements[k]==1)&&(disp_color==1)) n=2;
if((pinned_elements[k]==1)&&(disp_color==2)) n=-1;
l=old_coordinate[k]/DISPLAY_INCREMENT;
_setcolor(0);
_setpixel((160-5*MIDDLE_ELEMENT + 5*k),(100+l));
l=coordinate[k]/DISPLAY_INCREMENT;
_setcolor(disp_color+n);
_setpixel((160-5*MIDDLE_ELEMENT + 5*k),(100+l));
} /* DOFOR */
} /* IF */
   else {
l=(no_pendulums-40)/2;
first_element=l;
DOFOR(k,40) {
m=old_coordinate[l+k]/DISPLAY_INCREMENT;
n=0;
if((stability_element==(l+k))&&(stability_flag==1)) n=1;
if((pinned_elements[k]==1)&&(disp_color==1)) n=2;
if((pinned_elements[k]==1)&&(disp_color==2)) n=-1;
_setcolor(0);
_setpixel((60 + 5*k),(100+m));
m=coordinate[l+k]/DISPLAY_INCREMENT;
_setcolor(disp_color+n);
_setpixel((60 + 5*k),(100+m));
  } /* DOFOR */
}   /* ELSE */
}       /* DISPLAY_GRAPHICS */

display_text() {
/* The coordinate and other information below is a snapshot of what the
lattice is like at the particular instant in time when the key was pushed.*/
char message[80];
int c,j,k,l,m;
_setvideomode(_DEFAULTMODE);
printf("Time is : %lf",model_time);
printf("   System parameters are: \n");
printf("      Gamma         %lf",mean_gamma);
printf("      Eta          %lf",eta);
printf("      Omega         %lf\n",omega);
printf("      Beta          %lf",beta);
printf("      Alpha         %lf\n",alpha);
printf("\nThere are %d elements in the system",no_pendulums);
printf("\n\nPress any key to continue...");
c=getchar();
while(kbhit()==0) ;
printf("Element    Position    Velocity       Element    Position    Velocity\n");
DOFOR(j,20) {
```

```
printf("  %d      %lf      %lf          %d      %lf      %lf\n",
j,coordinate[first_element+j],momentum[first_element+j],
(j+20),coordinate[first_element+j+20],
momentum[first_element+j+20]);
} /* DOFOR */
printf("\n\nPress ^G for graphics, ^Q to quit.");
c=getchar();
while(kbhit()==0);
while(kbhit()==0);
}  /* display_text */

process_pause() {
/* Old method of saving files. Not sure if it has other uses.*/
   int c,i,j,k,l,mode;
   FILE *fr;
   char filename[30], message[80];
_clearscreen(_GCLEARSCREEN);
_setvideomode(_DEFAULTMODE);
printf("Do you wish to save this state?  ");
if((c=getche())==121) {
printf("\nEnter name of file to be written: ");
   scanf("%s",filename);
       if((fr=fopen(filename,"w"))!=NULL) {
  fprintf(fr,"%d\n",no_pendulums);
  fprintf(fr,"%lf\n",alpha);
  fprintf(fr,"%lf\n",beta);
  fprintf(fr,"%lf\n",eta);
  fprintf(fr,"%lf\n",omega);
  DOFOR(i,no_pendulums)
  fprintf(fr,"%lf  %lf  %lf  %lf\n",
omega0[i],gamma[i],old_coordinate[i],momentum[i]);
  fclose(fr);
} /* if() */
else  printf("Failed to open %s\n",filename);
} /* if (()) */
time_int=time_int*200/period;
printf("\nEnter new time multiple (old multiple is %lf):  ",time_int);
 scanf("%lf",&time_int);
 time_int=time_int*period/200;
 if(phase_flag==0) _setvideomode(_MRES4COLOR);
 else _setvideomode(_VRES16COLOR);
}  /* process_pause */

start_file_dump() {
/* Sets up a file dump of the coordinates of the selected element.  Set up
to record every time through for 30 times */
_clearscreen(_GCLEARSCREEN);
```

```
_setvideomode(_DEFAULTMODE);
printf("Enter number of element to be monitored: ");
scanf("%d",&chosen_element);
if(chosen_element > (no_pendulums-1))
printf("Out of range.  No file dump");
else file_dump_flag=1;
_clearscreen(_GCLEARSCREEN);
if(phase_flag==0)
_setvideomode(_MRES4COLOR);
else
_setvideomode(_VRES16COLOR);
counter1=0;
} /* start_file_dump */

stop_file_dump() {
/* Dumps the collected data to a file. */
char filename[30];
int c;
FILE *fs;
_clearscreen(_GCLEARSCREEN);
_setvideomode(_DEFAULTMODE);
printf("\nEnter name of file to be written: ");
scanf("%s",filename);
if((fs=fopen(filename,"w"))!=NULL) {
DOFOR(c,8000) fprintf(fs,"%lf %df\n",time_series_disp[c],gst[c]);
_clearscreen(_GCLEARSCREEN);
counter1=0;
if(phase_flag==0)
_setvideomode(_MRES4COLOR);
else _setvideomode(_VRES16COLOR);
} /* if(()) */
} /* stop_file_dump */

monitor_stability() {
/* Selects the lattice site to be monitored */
if(stability_flag==0) {
stability_flag=1;
disp_color=2;
_setvideomode(_DEFAULTMODE);
printf("Enter number of element to be monitored: ");
scanf("%d",&stability_element);
stability_restart  ();
}
else {
stability_flag=0;
disp_color=1;
}
```

```
if(phase_flag==0)
_setvideomode(_MRES4COLOR);
else
_setvideomode(_VRES16COLOR);
}

stability_check() {
/* The criterion is set for 1%. Appears to check to see if 20 consecutive
peaks are within the criterion of each other. */
int i,k;
k=0;
DOFOR(i,19) {
stable_flag=1;
if((peak_record[i+1]>(peak_record[i]*(1+STABILITY_INCREMENT)))|
(peak_record[i+1]<(peak_record[i]*(1 - STABILITY_INCREMENT ))))  {
stable_flag=0;
break;
} /* if */
}   /* DOFOR */
if(stable_flag==1) {
disp_color=1;
}
}

pin_elements() {
/* Pins an element at rest */
int ans,i,check;
_setvideomode(_DEFAULTMODE);
check=0;
printf("\nEnter number of element to be pinned:  ");
scanf("%d",&ans);
if((ans<0)|(ans>(no_pendulums-1))) {
check=1;
}
if(check==0) {
  if(pinned_elements[ans]==0) {
   pinned_elements[ans]=1;
coordinate[ans]=0;
momentum[ans]=0;
}
else pinned_elements[ans]=0;
} /* if */
if(phase_flag==0)
_setvideomode(_MRES4COLOR);
else
_setvideomode(_VRES16COLOR);
}
```

```
kick_elements() {
/* Equivalent to banging an element with a hammer.  No control over when
the hit occurs in the cycle. */
int ans,check;
double amount;
_setvideomode(_DEFAULTMODE);
check=0;
printf("\nEnter number of element to kick:  ");
scanf("%d",&ans);
if((ans<0)|(ans>(no_pendulums-1))) {
check=1;
}
if(check==0) {
printf("\n   Enter amount to kick:  ");
scanf("%lf",&amount);
coordinate[ans]=coordinate[ans]+amount;
} /* if */
if(phase_flag==0)
_setvideomode(_MRES4COLOR);
else
_setvideomode(_VRES16COLOR);
}

stability_restart() {
  /* Changes the color to show restart and zeros the stability check matrix */
int i;
char buff1[50];
disp_color=2;
stable_flag=0;
DOFOR(i,20) peak_record[i]=0;
peak_record[15]=10;
peak_amp=0;
sprintf(buff1,"Eta %.3lf  Omega %.3lf  Gamma %.2lf",eta,omega,mean_gamma);
 _moveto(10,10);
 _setfont("t'tms rmn'bn5");
 _outgtext(buff1);
}

save_state() {
 /* Automatic state saver.  It's supposed to keep the program from crashing
 when a lattice flys off numerically. */
    int i;
 char filename[]="savedsta";
    FILE *fl;
 /* change the number below to change strokes between autosaves */
 if(number_clicks == 100) {
  if(chdir("E:\\MODEL\\CLEON") != 0) {
```

```
screen_text();
printf("failed to change to E:\\MODEL\\CLEON.\n");
          stability_restart(); }
p_flag=0;
while(p_flag==0) {
 eqn_motion();
 }
if((fl=fopen(filename,"w")) != NULL) {
               fprintf(fl,"%d\n",no_pendulums);
               fprintf(fl,"%lf\n",alpha);
               fprintf(fl,"%lf\n",beta);
               fprintf(fl,"%lf\n",eta);
               fprintf(fl,"%lf\n",omega);
 DOFOR(i,no_pendulums)
  fprintf(fl,"%lf  %lf  %lf  %lf\n",
omega0[i],gamma[i],old_coordinate[i],old_momentum[i]);
 fclose(fl);
number_clicks=0; }
else  { screen_text();
                  printf("Failed to open %s\n",filename);
  stability_restart; } }
if(chdir("E:\\MODEL") != 0) {
screen_text();
printf("failed to change back to E:\\MODEL.\n");
stability_restart(); }
 else ++number_clicks;
 screen_graph(); }
    /* save_state */

save_stateu() {
 /* User initiated state save.  Normal way to save lattice data files.  Saves
 data for the lattice at a turning point (within one time step). */
    int i;
    char filename[30];
    FILE *fp;
    screen_text();
    printf("\nEnter name of file to save state in: ");
    scanf("%s",filename);
 p_flag=0;
  while(p_flag==0) {
 eqn_motion();
 }
  printf("\n%lf %lf",model_time,delta);
  if((fp=fopen(filename,"w")) != NULL) {
  fprintf(fp,"%d\n",no_pendulums);
        fprintf(fp,"%lf\n",alpha);
        fprintf(fp,"%lf\n",beta);
```

```
        fprintf(fp,"%lf\n",eta);
        fprintf(fp,"%lf\n",omega);
  DOFOR(i,no_pendulums)
            fprintf(fp,"%lf  %lf  %lf  %lf\n",
omega0[i],gamma[i],old_coordinate[i],old_momentum[i]);
  if(top_flag==1){
  fprintf(fp,"%lf  %lf  %lf  %lf\n",oldold_coordinate[stability_element],
old_coordinate[stability_element],coordinate[stability_element],time_int);
  fprintf(fp,"%lf  %lf  %lf",oldold_momentum[stability_element],
old_momentum[stability_element],momentum[stability_element]);}
  fclose(fp); }
  else  printf("Failed to open %s\n",filename);
 pause_flag=0;
printf("\nEnter number of steps per response cycle: ");
scanf("%d",&step_size);
if ((step_size%2) != 0) g=.5;
else g=0;
time_int=2*PI/(step_size*omega);
screen_graph(); }
    /*  save_stateu  */

retrieve_state() {

 /* Retrieves the auto saved file when the lattice crashes. */
 int i,c;
 char filename[]="savedsta";
    FILE *fg;
    screen_text();
    printf("\nLATTICE CRASH!!!!\n");
    printf("Do you want to retrieve the latest stored state(Y/N).\n");
 printf("  Eta  %lf  Omega  %lf\n",eta, omega);
 if((c=getche())==121) {
 if(chdir("E:\\MODEL\\CLEON") != 0) {
printf("Failed to change directories.\n");
}
}
else { printf("\nEnter name of file to retrieve.: ");
          scanf("%s",filename); }
if((fg=fopen(filename,"r"))!=NULL) {
           mean_gamma=0;
fscanf(fg,"%d\n",&no_pendulums);
fscanf(fg,"%lf\n",&alpha);
fscanf(fg,"%lf\n",&beta);
fscanf(fg,"%lf\n",&eta);
fscanf(fg,"%lf\n",&omega);
DOFOR(i,no_pendulums) {
 fscanf(fg,"%lf  %lf  %lf  %lf\n",
```

```
                &omega0[i],&gamma[i],&coordinate[i],&momentum[i]);
                mean_gamma=mean_gamma+gamma[i];
}
fclose(fg);
counter2=0;
delta=atan((omega*beta)/sqrt(SQR(eta)-(SQR(omega)*SQR(beta))))/2;
model_time=0.0;
}
else printf("Can't open file requested.");
time_int=time_int*200/period;
printf("\nEnter new time multiple (old multiple is %lf):  ",time_int);
scanf("%lf",&time_int);
time_int=time_int*period/200;
if(chdir("E:\\MODEL") != 0) {
screen_text();
printf("failed to change back to E:\\MODEL.\n"); }
screen_graph();
}
 /* retrieve_state */

screen_text() {
    _clearscreen(_GCLEARSCREEN);
    text_flag=1;
    graphics_flag=0;
    spectrum_flag=0;
    _setvideomode(_DEFAULTMODE); }
    /* screen_text  */

screen_graph() {
    _clearscreen(_GCLEARSCREEN);
    _setvideomode(_MRES4COLOR);
    _setcolor(1);
    phase_flag=0;
    spectrum_flag=0;
    text_flag=0;
    graphics_flag=1; }
    /*  screen_graph  */

linear_kick() {
  /* Lattice perturbation up to a maximum of 3%.  Sort of random as to sign of
kick direction for each site.  Uses the % of the maximum amplitude site. */
int a,j,i;
double k,b;
struct dostime_t time2;
_dos_gettime(&time2);
srand(time2.hsecond);
DOFOR(i,no_pendulums) {
```

```
 b=rand();
 /* change 33.33 to change %, now 3% */
 k=b/(33.33 * 32767);
 amp_kick=k*peak_amp;
 if(b > 16384) {
 coordinate[i] = coordinate[i] +amp_kick;
 }
 else coordinate[i] = coordinate[i] - amp_kick;
 }
 peak_amp=0.0;
 } /* linear_kick */

time_wait() {
 /* part of a routine to get the lattice at the turning point. */
t_flag=0;
 while(t_flag==0){
  eqn_motion();
  }
  }/* time_wait */

init_phasplot() {
/* Sets up the phase plot to look at the phase plane of sites. */
int c,i,j,k,l;
char ans[5],message[40],txt[3];
float dummy;

spectrum_flag=0;
_setvideomode(_DEFAULTMODE);
DOFOR(i,5) phase_elements[i]=0;
printf("\nDo you want Poincare sections?  ");
if((c=getch())==121) phase_flag=2;
else phase_flag=1;
printf("\nWhich elements do you wish to monitor (999 to finish): ");
i=-1;
k=0;
while((i++!=999)&&(k<5)) {
scanf("%d",&i);
phase_elements[k++]=i+1;
} /* while */
DOFOR(i,5) {
if(phase_elements[i]==1000) {
phase_elements[i]=0;
}
}
_setvideomode(_VRES16COLOR);
_clearscreen(_GCLEARSCREEN);
graphics_flag=0;
```

```
_settextcolor(7);
colors[0]=2;
colors[1]=3;
colors[2]=9;
colors[3]=14;
colors[4]=13;
 DOFOR(i,120) {
_setpixel(318,4*i);
}
DOFOR(i,160) {
_setpixel(4*i,238);
}
DOFOR(i,6) {
_setpixel(319,10*SCREEN_CORRECTION_FACTOR+1
+(80/SCREEN_CORRECTION_FACTOR)*(i+1));
}
DOFOR(i,8) {
_setpixel(80*(i+1),239);
}
_moveto(480,450);
DOFOR(i,5) {
if((phase_elements[i])!=0) {
c=phase_elements[i]-1;
  /*       itoa(c,txt,10);
_outtext(txt); */
printf("%d  ",c);
_setcolor(colors[i]);
DOFOR(j,4) {
DOFOR(k,8) {
_setpixel(550+10*i+k,10+j);
}
}
}
}  /*  DOFOR */
printf("     LAT2X486.C");
printf("\nGamma:  %lf  Eta:  %lf  Beta:  %lf",mean_gamma,eta,beta);
printf("\nOmega:  %lf  Alpha:  %lf   Time interval:  %lf",
omega,alpha,time_int);
dummy=1/(time_int*omega);
phase_int=dummy/1;
phase_counter=0;
}

phasplot() {
/* Displays the phase for chosen elements. */
int i,j,k,l;
double speed, position,fast_coordinate,fast_momentum;
```

```
if(phase_flag==1) {
DOFOR(i,5) {
if((j=phase_elements[i]-1)!=-1) {
speed=PHASE_INCREMENT*momentum[j];
_setcolor(colors[i]);
position=PHASE_INCREMENT*coordinate[j];
fast_coordinate=(position*cos(omega*model_time)
-speed*sin(omega*model_time)/omega)
/(3*SCREEN_CORRECTION_FACTOR);
fast_momentum=(position*sin(omega*model_time)
+speed*cos(omega*model_time)/omega)/4;
_setpixel((320+320*fast_coordinate),(240+240*fast_momentum));
} /* if() */
}   /* DOFOR */
}     /* if */
else if(phase_flag==2) {
 if(phase_counter==phase_int) {
phase_counter=0;
DOFOR(i,5) {
if((j=phase_elements[i]-1)!=-1) {
speed=PHASE_INCREMENT*momentum[j];
_setcolor(colors[i]);
position=PHASE_INCREMENT*coordinate[j];
fast_coordinate=(position*cos(omega*model_time)
-speed*sin(omega*model_time)/omega);
fast_momentum=(position*sin(omega*model_time)
+speed*cos(omega*model_time)/omega)/4;
_setpixel((320+320*fast_coordinate),(240+240*fast_momentum));
} /* if() */
}   /* DOFOR */
 } /* if */
} /* else if() */
} /* phasplot */

/* FFT Routines for lattice programs */

compute_spectrum(){ /* Uses algorithm from p. 411 of Numerical Recipes in C */
int n,mmax,m,j,istep,i,nn,temp;
double wtemp,wr,wpr,wpi,wi,theta,tempr,tempi;
nn=2048;
n=nn<<1;
j=1;
DOFOR(i,2048) {
nn=(i&1024)/1024;
nn=nn+(i&512)/256;
nn=nn+(i&256)/64;
nn=nn+(i&128)/16;
```

```
nn=nn+(i&64)/4;
nn=nn+(i&32);
nn=nn+(i&16)*4;
nn=nn+(i&8)*16;
nn=nn+(i&4)*64;
nn=nn+(i&2)*256;
nn=nn+(i&1)*1024;
temp2[2*nn]=spectrum[2*i];
temp2[2*nn+1]=spectrum[2*i+1];
}
DOFOR(i,4096) spectrum[i]=temp2[i];
mmax=2;
while(n>mmax) {
istep=2*mmax;
theta=2*PI/mmax;
wtemp=sin(0.5*theta);
wpr=-2.0*wtemp*wtemp;
wpi=sin(theta);
wr=1.0;
wi=0.0;
for(m=0;m<(mmax-1);m+=2) {
for(i=m;i<=n;i+=istep) {
j=i+mmax;
tempr=wr*spectrum[j]-wi*spectrum[j+1];
tempi=wr*spectrum[j+1]+wi*spectrum[j];
spectrum[j]=spectrum[i]-tempr;
spectrum[j+1]=spectrum[i+1]-tempi;
spectrum[i]+=tempr;
spectrum[i+1]+=tempi;
}
wr=(wtemp=wr)*wpr-wi*wpi+wr;
wi=wi*wpr+wtemp*wpi+wi;
}
mmax=istep;
}
}

plot_spectrum() {
int i,j;
double freq,magnitude,mag,max;
int c;
FILE *fp;
if(dump_spectrum_flag==1) {
printf("\nDumping spectrum now");
fp=fopen("spectrum.out","w");
for(i=1;i<2048;i++) {
if(time_int!=0) freq=i/(2048*time_int);
```

```
else printf("Time_int was zero!");
magnitude=sqrt(spectrum[(2*i)]*spectrum[(2*i)]
+spectrum[(2*i+1)]*spectrum[(2*i+1)]);
fprintf(fp,"%lf  %lf  \n",freq,magnitude);
}
fclose(fp);
dump_spectrum_flag=0;
}
_clearscreen(_GCLEARSCREEN);
max=0;
for(i=1;i<2049;i++) { /* this loop calculates the max spectral component*/
magnitude=sqrt(spectrum[(2*(i))]*spectrum[(2*(i))]
+spectrum[(2*(i)+1)]*spectrum[(2*(i)+1)]);
if(magnitude>max) max=magnitude;
}
_setcolor(3);
DOFOR(i,640) _setpixel(i,460);
DOFOR(i,11) _setpixel(51*i,461),_setpixel(51*i,462);
printf("Spectrum for element %d     LATTIC2X.C",spectrum_element);
printf("\nAlpha: %lf Beta: %lf Gamma: %lf Eta: %lf Omega: %lf",
alpha,beta,mean_gamma,eta,omega);
printf("\nMax amp = %lf     ",max);
freq=512/(2048*time_int);
printf("Max freq shown: %lf     Time interval:  %lf",freq,time_int);
_setcolor(1);
DOFOR(i,512) {
j=0;
magnitude=sqrt(spectrum[2*i]*spectrum[2*i]
+spectrum[2*i+1]*spectrum[2*i+1]);
mag=magnitude*400/max;
while(j++<mag) {
_setpixel(i,460-j);
} /* while */
}  /* DOFOR */
}

init_waterfall() {
/* Sets up waterfall type display to see trends in latice motion. Good for
observing kink or breather drift. */
int c,i,j,k,l,m;
char ans[5];
_setvideomode(_VRES16COLOR);
_setcolor(1);
DOFOR(i,600) _setpixel(20+i,20);
DOFOR(i,450) _setpixel(20,i+20);
DOFOR(i,61) {
if((i==10)|(i==20)|(i==32)|(i==42)|(i==52)) _setcolor(2);
```

```
DOFOR(j,113) {
_setpixel(20+10*i,20+4*j);
}
_setcolor(1);
}
_setcolor(2);
DOFOR(i,11) {
DOFOR(j,160) {
_setpixel(20+4*j,16+40*(i+1));
}
}
waterfall_counter=0;
color_counter=2;
}

disp_waterfall() {
int c,i,j,k,l;
if(waterfall_counter++<110) {
if(color_counter+++==14) {
color_counter=2;
_setcolor(color_counter);
}
else {
_setcolor(color_counter);
}
DOFOR(i,no_pendulums) {
 if(waterfall_counter<55) {
_setpixel(20+2*i,24+8*waterfall_counter-7*wat_inc*(coordinate[i]));
 }
 else {
_setpixel(340+2*i,24+8*(waterfall_counter-55)-7*wat_inc*(coordinate[i]));
}
}
}
else {
wait_flag=1;
}
}

eqn_motion() {
/*    The following loop calculates a round of velocities for one
iteration of the model clock.  This is where the equations
of motion need to be located, if one wishes to change the
model's physics!  */
int i,j,k;
 DOFOR(i,no_pendulums)
if(coordinate[i] >100) {
```

```
retrieve_state();
}
  DOFOR(i,no_pendulums) {
if(i==0) {
acceleration=gamma[i]*(coordinate[i+1]+coordinate[no_pendulums-1]
-2*coordinate[i])-beta*momentum[i]
-(SQR(omega0[i])+2*eta*cos(2*omega*model_time-delta))*coordinate[i]
+alpha*CUB(coordinate[i]);
}
else if(i==(no_pendulums-1)) {
acceleration=gamma[i]*(coordinate[i-1]+coordinate[0]
-2*coordinate[i])-beta*momentum[i]
-(SQR(omega0[i])+2*eta*cos(2*omega*model_time-delta))*coordinate[i]
+alpha*CUB(coordinate[i]);
}
else {
acceleration=gamma[i]*(coordinate[i+1]+coordinate[i-1]
-2*coordinate[i])-beta*momentum[i]
-(SQR(omega0[i])+2*eta*cos(2*omega*model_time-delta))*coordinate[i]
+alpha*CUB(coordinate[i]);
}
oldold_momentum[i]=old_momentum[i];
old_momentum[i]=momentum[i];
momentum[i]=momentum[i]+acceleration*time_int;
}  /* DOFOR */
  DOFOR(i,no_pendulums) {
oldold_coordinate[i]=old_coordinate[i];
old_coordinate[i]=coordinate[i];
if(pinned_elements[i]==0) {
coordinate[i]=coordinate[i]+momentum[i]*time_int;
}

/* Records data for file dump */
if((i==chosen_element)&&(file_dump_flag==1)) {
 /*  if(counter2++==30) { */
time_series_disp[counter1++]=coordinate[i];
gst[counter1++]=g;
 /*     counter2=0;
}*/
if(counter1==8000) stop_file_dump();
} /* if((i... */

/* Stores energy information */
if(energy_flag==1) {
energy=energy+0.5*(SQR(momentum[i])
+ gamma[i]*SQR((coordinate[i+1]-coordinate[i]))
+SQR(coordinate[i]))-alpha/4*pow(coordinate[i],4);
```

```
} /* if(energy...) */

/* Stores FFT information */
if((spectrum_flag==1)&&(spectrum_element==i)
&&((sample_counter++)==1)) {
if(spectrum_counter<2048) {
spectrum[2*spectrum_counter]=coordinate[i];
spectrum[2*spectrum_counter+1]=0;
spectrum_counter++;
sample_counter=0;
}
else {
compute_spectrum();
_clearscreen(_GCLEARSCREEN);
_setvideomode(_VRES16COLOR);
plot_spectrum();
phase_flag=0;
text_flag=0;
graphics_flag=0;
DOFOR(j,4096) spectrum[j]=0;
spectrum_counter=0;
sample_counter=0;
} /* else */
} /* if ((spec... */

/* Records the peaks for the stability checker */
if(((i==stability_element)&&(stability_flag==1)&&(stable_flag==0))|
((i==(no_pendulums-1))&&(waterfall_flag==1))) {
if((coordinate[i]>0)&&(coordinate[i]<old_coordinate[i])
&&(peak_flag==0)) {
peak_flag=1;
if(pause_flag==1) pause_flag2=1;
DOFOR(k,19) peak_record[k]=peak_record[k+1];
peak_record[19]=coordinate[i];
if(stability_flag==1) stability_check();
if(waterfall_flag==1) disp_waterfall();
}
else if(coordinate[i]<0) peak_flag=0;
}

/* Records peak amplitude for display and linear kick */
if((coordinate[i] > 0)&&(coordinate[i] < old_coordinate[i])) {
if(old_coordinate[i]>peak_amp){
peak_amp = old_coordinate[i];
}
}
if((i==stability_element)&&(stability_flag==1)&&(stable_flag==1)){
```

```
 if(((coordinate[i]-old_coordinate[i]) > = 0) &&
 ((old_coordinate[i]-oldold_coordinate[i]) < = 0) && (coordinate[i] < 0)){
p_flag=1;
}
}

} /* DOFOR */

/* Increments time and resets it at 2Pi.  Causes problems in eqns of
motion if not reset */
model_time=model_time+time_int;
if((g+1) == (step_size/2)){
model_time=0.0;
t_flag=1;
g=0;
}
else g++;

/* Displays energy information gathered above. */
if(graphics_flag==1) display_graphics();
if((text_flag==1)&&(energy_flag==1)) {
printf("\n Total Energy at time %lf is %lf",model_time,energy);
energy_counter++;
if(energy_counter==20) {
printf("\n\nStrike any key to continue...");
while(kbhit()==0);
energy_counter=0;
}
}
if(phase_flag!=0) {
phasplot();
phase_counter++;     /* Used for Poincare sections */
}
}
```

2. Upper Cutoff, Global Drive, Equation of Motion

```
        eqn_motion() {
/*   The following loop calculates a round of velocities for one
iteration of the model clock.  This is where the equations
of motion need to be located, if one wishes to change the
model's physics!  */
int i,j,k;
 DOFOR(i,no_pendulums)
if(coordinate[i] > 100) {
retrieve_state();
```

```
}
  DOFOR(i,no_pendulums) {
if(i==0) {
acceleration=gamma[i]*(coordinate[i+1]+coordinate[no_pendulums-1]
-2*coordinate[i])-beta*momentum[i]
-(SQR(omega0[i])+2*eta*cos(2*omega*model_time-delta))*coordinate[i]
+alpha*CUB(coordinate[i]);
}
else if(i==(no_pendulums-1)) {
acceleration=gamma[i]*(coordinate[i-1]+coordinate[0]
-2*coordinate[i])-beta*momentum[i]
-(SQR(omega0[i])+2*eta*cos(2*omega*model_time-delta))*coordinate[i]
+alpha*CUB(coordinate[i]);
}
else {
acceleration=gamma[i]*(coordinate[i+1]+coordinate[i-1]
-2*coordinate[i])-beta*momentum[i]
-(SQR(omega0[i])+2*eta*cos(2*omega*model_time-delta))*coordinate[i]
+alpha*CUB(coordinate[i]);
}
old_momentum[i]=momentum[i];
momentum[i]=momentum[i]+acceleration*time_int;
}  /* DOFOR */
  DOFOR(i,no_pendulums) {
oldold_coordinate[i]=old_coordinate[i];
old_coordinate[i]=coordinate[i];
if(pinned_elements[i]==0) {
coordinate[i]=coordinate[i]+momentum[i]*time_int;
}
```

3. Lower Cutoff, End Driven, Equation of Motion

```
        eqn_motion() {
/*   The following loop calculates a round of velocities for one
iteration of the model clock.  This is where the equations
of motion need to be located, if one wishes to change the
model's physics!  */
int i,j,k;
DOFOR(i,no_pendulums)
if(coordinate[i] > 1000) {
retrieve_state();
}
  DOFOR(i,no_pendulums) {
if(i==0) {
acceleration=gamma[0]*(coordinate[1]-coordinate[0])
-beta*momentum[0]
```

```
-((coordinate[i]/sqrt(1 + 2*SQR(coordinate[i]))) +eta*cos(omega*model_time));
  }
else if(i==no_pendulums-1){
acceleration=gamma[i]*(coordinate[i-1]-coordinate[i])
-beta*momentum[i]
-(coordinate[i]/sqrt(1 + 2*SQR(coordinate[i])));
  }
else {
acceleration=gamma[i]*(coordinate[i+1]+coordinate[i-1]
-2*coordinate[i])-beta*momentum[i]
-(coordinate[i]/sqrt(1 + 2*SQR(coordinate[i])));
  }
old_momentum[i]=momentum[i];
momentum[i]=momentum[i]+acceleration*time_int;
} /* DOFOR */
  DOFOR(i,no_pendulums) {
old_coordinate[i]=coordinate[i];
if(pinned_elements[i]==0) {
coordinate[i]=coordinate[i]+momentum[i]*time_int;
}
```

4. Upper Cutoff, End Driven, Equation of Motion

```
        eqn_motion() {
/*   The following loop calculates a round of velocities for one
iteration of the model clock.  This is where the equations
of motion need to be located, if one wishes to change the
model's physics!  */
int i,j,k;
DOFOR(i,no_pendulums)
if(coordinate[i] >100) {
retrieve_state();
}
  DOFOR(i,no_pendulums) {
if(i==0) {
acceleration=gamma[0]*(coordinate[1]-coordinate[0])
-beta*momentum[0]
-(SQR(omega0[0])*coordinate[0]+eta*cos(omega*model_time))+alpha*CUB(coordinate[0]);
}
else if(i==no_pendulums-1){
acceleration=gamma[i]*(coordinate[i-1]-coordinate[i])
-beta*momentum[i]
-(SQR(omega0[i])*coordinate[i])+alpha*CUB(coordinate[i]);
}
else {
acceleration=gamma[i]*(coordinate[i+1]+coordinate[i-1]
```

```
-2*coordinate[i])-beta*momentum[i]
 -(SQR(omega0[i])*coordinate[i])+alpha*CUB(coordinate[i]);
}
old_momentum[i]=momentum[i];
momentum[i]=momentum[i]+acceleration*time_int;
}  /* DOFOR */
  DOFOR(i,no_pendulums) {
old_coordinate[i]=coordinate[i];
if(pinned_elements[i]==0) {
coordinate[i]=coordinate[i]+momentum[i]*time_int;
}
```

LIST OF REFERENCES

Atchley, Mary, work in progress, December 1991.

Denardo, Bruce C., *Observations of Nonpropagating Oscillatory Solitons*, Ph. D. Dissertation, UCLA, Los Angeles, California, January 1990.

Denardo, B., Galvin, B., Greenfield, A., Larazza, A., Putterman, S. and Wright, W., "Observations of Localized Structures in Nonlinear Vibratory Lattices: Domain Walls and Kinks," in revision Phys. Rev. Lett., December 1991.

Dodd, R., Eilbeck, J., Gibbon, J. and Morris, H., *Solitons and Nonlinear Wave Equations*, Academic Press, 1982.

Galvin, Brian R., *Numerical Studies of Standing Solitons in a Nonlinear Lattice*, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1991.

Goedde, C.G., Lichtenberg, A.J., and Lieberman, M.A., "Parametric Instabilities in the Discrete Sine-Gordon Equation," Physica, v. D 41, pp. 341-355, 1990.

Kit, E., Shemer, L., and Miloh, T., "Experimental and Theoretical Investigation of Nonlinear Sloshing Waves in a Rectangular Channel," J. Fluid Mech., v. 181, pp. 265-291, 1987.

Mollenauer, L. F., Gordon, J. P., and Evangelides S.G., "Multigigabit Soliton Transmissions Transverse Ultralong Distances," Laser Focus World, pp. 159-170, November 1991.

Segur, H., and Kruskal, M., "Nonexistence of Small-amplitude Breather Solutions in the ϕ^4 Theory," Phys. Rev. Lett., v. 58, pp. 747-750, 1987.

Shemer, L., "On the Directly Generated Resonant Standing Waves in a Rectangular Tank," J. Fluid Mech., v. 217, pp. 143-165, 1990.

Wu, J., Keolian, R., and Rudnick, I., "Observation of a Nonpropagating Hydrodynamic Soliton," Phys. Rev. Lett., v. 52, pp. 1421-1424, 1984.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 052 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Bruce Denardo Physics Department, PH-De Naval Postgraduate School Monterey, California 93943-5002	3
4.	Andrés Larraza Physics Department, PH-La Naval Postgraduate School Monterey, California 93943-5002	3
5.	Lieutenant Cleon A. Walden, Jr., USN c/o Cleon A. Walden, Sr. 1131 Rocky Brook Road Cedar Hill, Texas 75104	2
6.	Professor Karlheinz Woehler, Chairman PH Physics Department Naval Postgraduate School Monterey, California 93943-5000	1
7.	Professor Steven Garrett Physics Department, Ph-Ga Naval Postgraduate School Monterey, California 93943-5002	1